CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

A MEDIUM-BAND SPEECH CODING SCHEME USING VECTOR QUANTIZATION

by

kin-shing Yu

A thesis
presented to the School of
Graduate Studies and Research
of the University of Ottawa
in partial fulfillment of the requirements
for the degree of
Master of Applied Sciences
in Electrical Engineering

Ottawa, Canada, 1986

ISBN 0-315-36562-5

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below and give address and date.

ABSTRACT

The basic method of the full search vector quantizer (VQ) is reviewed. The design procedure of a VQ is further discussed.

The performance of the VPCM is investigated. Based upon the VPCM, a medium-band speech coding system is developed. This coding system includes a redundancy reducing technique, an adaptive technique and a 256-codewords 8-dimensional VPCM. The redundancy reducing technique which makes use of the cycle to cycle and the pitch period to pitch period dependencies of the speech waveform is introduced here. The adaptive technique which makes use of the statistical charateristics of the speech waveform is based upon the adaptive technique developed by Cuperman.

Finally, the computation complexity of the developed coding system is investigated and a hardware architecture for the encoder is proposed.

ACKNOWLEDGEMENT

The author wishes to express his sincere gratitude to his academic supervisor Dr. Willem Steenaart for his patient guidance and moral support during the course of the program.

Special thanks is extended to Dr. Melvyn J. Hunt, a research officier in the National Research Council of Canada, for his generosity in providing the real speech data to make this research work possible.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 General review of vector quantization

Vector Quantization, a powerful coding technique, is a promising tool that allows substantial transmission bit rate reduction in speech coding. It was first applied in Linear Predictive Coding [4] and, it has allowed a 2400 bits/sec pitch-excited linear predictive coder to reduce the transmission bit rate to only 800 bits/sec with very slight speech quality reduction [9]. In [8], vector quantization is applied in residually-excited linear predictive coding (RELP), the 13800 bits/sec RELP was found to provide a more natural speech sound than the 15000 bits/sec adaptive predictive coder (APC) [21].

In waveform coding, the direct application of vector quantization to the sampled speech waveform, a direct generalization of the conventional linear PCM called as the Vector PCM (VPCM), reported in [15] shows that VPCM results in higher signal to quantization noise ratio (SNR) than that of DPCM and ADPCM at both 1 bit/sample (8000 bits/sec) and 2 bits/sample (16000 bits/sec). A similar study on VPCM [6] reported that several popular speech waveform coders such as the Fourier transform coder and the subband coder are both inferior to VPCM, in the sense of lower SNR, for around 2 bits/sample.

Vector quantization is a block source coding scheme that considers a source vector as a single entity and encodes all of the vector components at the same time. The encoding process of such a block quantizer is to select an reproduction vector (codeword) from a stored finite set of reproduction vectors (codebook) for each source vector. A binary word that identifies the selected codeword is used in transmission. The decoding process is simply a table look-up procedure that converts the received binary word into the codeword which was selected in the encoding process.

The performance of this block source coder is predicted by the rate-distortion theory that the distortion-rate function is achievable in the limit when the vector dimension goes to infinity [2]. However, the rate-distortion theory does not offer any design method for this block coder. Therefore, vector quantization did not receive so much attention as scalar quantization schemes during the past decade [2,5,6].

In 1978, Y. Linde, A. Buzo and R.M. Gray developed an iterative algorithm for vector quantizer design [5], which is found to be useful and is extensively used. This algorithm is sometimes called the LBG algorithm. As a matter of fact, most of the research in vector quantization are conducted after the LBG algorithm was introduced.

Recently, a careful study [2] has shown that vector quantization is more efficient in redundancy removal than scalar quantization. In [2], vector quantization is

described as a redundancy removal process that makes effective use of four properties of the vector components: 1) the correlation and 2) the non-linear dependency (not correlated but dependent) among the vector components, 3) the shape of the probability density function of the vector components and 4) the vector dimension. Scalar quantization together with vector rotation (decorrelating transformation) and bit allocation, however, can utilize effectively only the correlation among the vector components and the shape of the probability density function of the vector components [2]. These advantages of vector quantization with finite vector dimension over scalar quantization are mainly the reason why vector quantization with small vector dimension is so promising in the speech coding research area.

As mentioned previously, vector quantization is better than scalar quantization in the sense of being more efficient in redundancy removal with smaller quantization error resulting. However, to account for the computation complexity and memory storage requirement, vector quantization is much more demanding than scalar quantization. The required computation complexity for the encoding process and the memory storage for the codebook grows exponentially with the product of the vector dimension and the number of codewords. Since the capacity of read-only memories is increasing and their cost is decreasing, the main obstacle in using vector quantization is the high computation complexity.

Several attempts have been made to reduce the computation complexity requirement. The approaches being taken fall into two general categories : The first is to use specially structured codebooks to reduce the computation requirement. Tree search vector quantization [1,2,3,4,7,9] and product code vector quantization [1,2,3,4,10] are typical examples. However, the structured codebooks always lead to suboptimal vector quantizers. The structured codebooks may not be optimal and the encoding process, based on the structured codebooks, may not be able to select the best codeword available. In the second approach , a modified encoding process which requires much less computation than the original encoding process is used. Since the codebook of the original vector quantizer is used, there is no performance degradation resulting from the modified encoding process. The modified encoding processes are termed as the fast search algorithms [26].

1.2 Thesis objective

Vector quantization together with a fast search algorithm certainly offers a new and promising research direction in speech coding. The objective of the research work for the thesis is to develop an implementable medium-band speech coder, with at least communication quality output speech, based upon vector quantization. Since vector quantization can be applied to an arbitrary source, it is applicable either for waveform coding or for voice coding. Unfortunately, voice coding usually requires complicated

computation (see chapter 2, section 2.4.2). The application of vector quantization in voice coding schemes does not guarantee the developed speech coder is real-time implementable with today's technologies. To my knowledge, only the real-time implementation of VPCM is reported in the literature [24].

Therefore, to account for the implementation constraint, it was decided to make the projected speech coder a speech waveform coder. VPCM is, therefore, selected as the basic coding scheme of the required speech coder.

The performance of VPCM is not competitive with that achieved by some sophisticated scalar speech compression techniques such as the Adaptive Transform Coding (ATC). However, the success of vector quantization in linear predictive coding suggests that by transforming the source before vector quantization results in performance improvement. The purpose of the transformation usually involves certain redundant information removal. The basic idea of using such a transformation is to share the "required task" that needs to be done by a certain quantizer. For instance, the discrete cosine transformation (decorrelating transformation) in ATC reduces the correlation among veetor components so that the corresponding quantizer does not need to to be designed utilizing any correlation among vector components.

Another method in performance improvement is to divide the speech signal into several different classes, subject to

different statistical characteristics, and use different quantizers for different classes of speech. The adaptive vector quantizer [3] studied by Gersho and Cuperman shows that a 2.5 dB improvement in SNR at 1 bit/sample is achieved over the non-adaptive vector quantizer. This adaptive technique can be combined with certain redundant information reducing techniques to form a new and powerful coding scheme. The adaptive vector predictive coding (AVPC) developed by Cuperman [22] demonstrated that by combining the adaptive technique and the vector linear prediction (redundant information removal) with vector quantization, a segmental SNR (SEGSNR) of about 20 dB is obtained at 2 bits/sample.

The final speech coder of interest, therefore, is a VPCM which employs certain redundant information removal and adaptive techniques. The output speech quality is judged by means of the SNR and the segmental SNR (SEGSNR) while the computation complexity is estimated by means of the number of operations (multiplication, addition, comparison, etc) per sample.

1.3 Thesis organization

A general review of vector quantization is given in chapter 2. It starts with the basic concepts and the performance parameters of the full search vector quantizer. The design method of an arbitrary full search vector quantizer is discussed next. Chapter 2 ends with the design procedure of the full search vector quantizer.

Chapter 3 starts with the general review of VPCM and the redundant information of the speech waveform. A simple but efficient redundant information reducing technique for VPCM is then introduced using an adaptive scaling of the speech signal. In addition, the structure and design method of VPCM together with the adaptive scaling scheme is further discussed in detail. Afterwards, an adaptive VPCM which is based upon the adaptive technique developed by Gersho and Cuperman is introduced and discussed. Finally, the adaptive scaling scheme together with the adaptive VPCM is combined as the final speech coder. Subsequently, a comparison among all the different extensions of VPCM is given with their output speech quality and transmission bit rate.

Chapter 4 is basically a discussion of the computation complexities of the different extensions of VPCM discussed in chapter 3. A proposed hardware architecture for the desired speech coder is also discussed.

CHAPTER 2

INTRODUCTION TO VECTOR QUANTIZATION

2.1 Introduction

During the past few years, a number of different vector quantizers have been developed [1]. Out of these vector quantizers, memoryless vector quantizers are now used in speech coding, image coding and speech recognition. Memory vector quantizer such as Finite State Vector Quantizer (FSVQ) may be another active candidate in speech coding for the next few years [11]. Nevertheless, the memoryless vector quantizer, if subject to very long (infinite) block length does not perform worse than the memory vector quantizer. If used with certain fast search algorithms [26] developed recently, the memoryless vector quantizer will not become obsolete.

In this chapter, only the basic memoryless vector quantizer is discussed. This basic memoryless vector quantizer is known in the literature as the full search vector quantizer (VQ).

The material covered in this chapter, except for section 2.6 (the VQ design procedure) is also applicable to some other memoryless vector quantizers such as the Tree Search Vector Quantizer (TSVQ). The basic concept of a VQ is introduced in section 2.2. Section 2.3 covers the performance parameters of a VQ. The design of a VQ will be given in section 2.5 followed by the discussion of the distortion measures for speech coding systems in section

2.4. Afterwards, a design procedure of a VQ, subject to a particular transmission bit rate, is given in section 2.6.

2.2 Full search vector quantizer (VQ)

A N-level (N codewords), k-dimensional vector quantizer is a source coder that consists of a coder and a decoder. The coder consists of a codebook, $Y=\{\underline{y}_n: n=1,\ldots,N\}$, where $\underline{y}_n$ is a k-dimensional vector, and an index set, $J=(1,\ldots,N)$, which completely identifies each $\underline{y}_n$ in Y. The decoder consists of the same codebook Y and the same index set J as the coder.

As in any other quantization process, a non-negative distortion measure $d(\underline{x},\underline{y})$ between the input vector $\underline{x}$ and the codeword $\underline{y}$ is necessary. Based upon a given distortion measure, the encoding process of the vector quantizer can be separated into two steps.

Step 1) The coder assigns a codeword $\underline{y}_n$ to each input vector $\underline{x}$ or $Q_1(\underline{x})=\underline{y}_n$ subject to

$$d(\underline{x},\underline{y}_n) \leq d(\underline{x},\underline{y}_m) \quad \text{for all } n \neq m$$

if several codewords have the same minimum distortion with respect to $\underline{x}$, then simply assign the codeword that has the lowest index n [14]. The codeword $\underline{y}_n$ selected in this manner is called the "nearest neighbor" of $\underline{x}$.

Step 2) The coder maps the selected codeword into the index set J, that is, $Q_2(y_n)=n \in J$. The index set J, which may be called the "codebook address", is then transmitted to the decoder.

The decoding process of the vector quantizer is simply the reverse process of step 2) above. Thus, the output of the decoder is the codeword selected by the coder. This basic structure of vector quantization is shown in Fig.2.1. If the coder must search all N codewords in step 1), then the vector quantizer is named a full search vector quantizer (VQ).

Since only the index set J is transmitted, the transmission rate, r, is then given by

$$r = \frac{\text{wordlength of } J}{k} \quad \text{bits/dimension}$$

If N is a power of 2, then the wordlength of J is $\log_2 N$ bits. Thus,

$$r = \frac{\log_2 N}{k} \quad \text{bits/dimension} \qquad (1)$$

Subject to the input vector rate F_s in dimensions/sec, the transmission bit rate is,

$$R = \frac{\log_2 N}{k} \cdot F_s \quad \text{bits/sec}$$

Note that as $k \to \infty$, $r \to 0$. Extremely low transmission rates can be achieved by making use of very long block length. However, a very long block length is not practical as the computation complexity of VQ grows exponentially

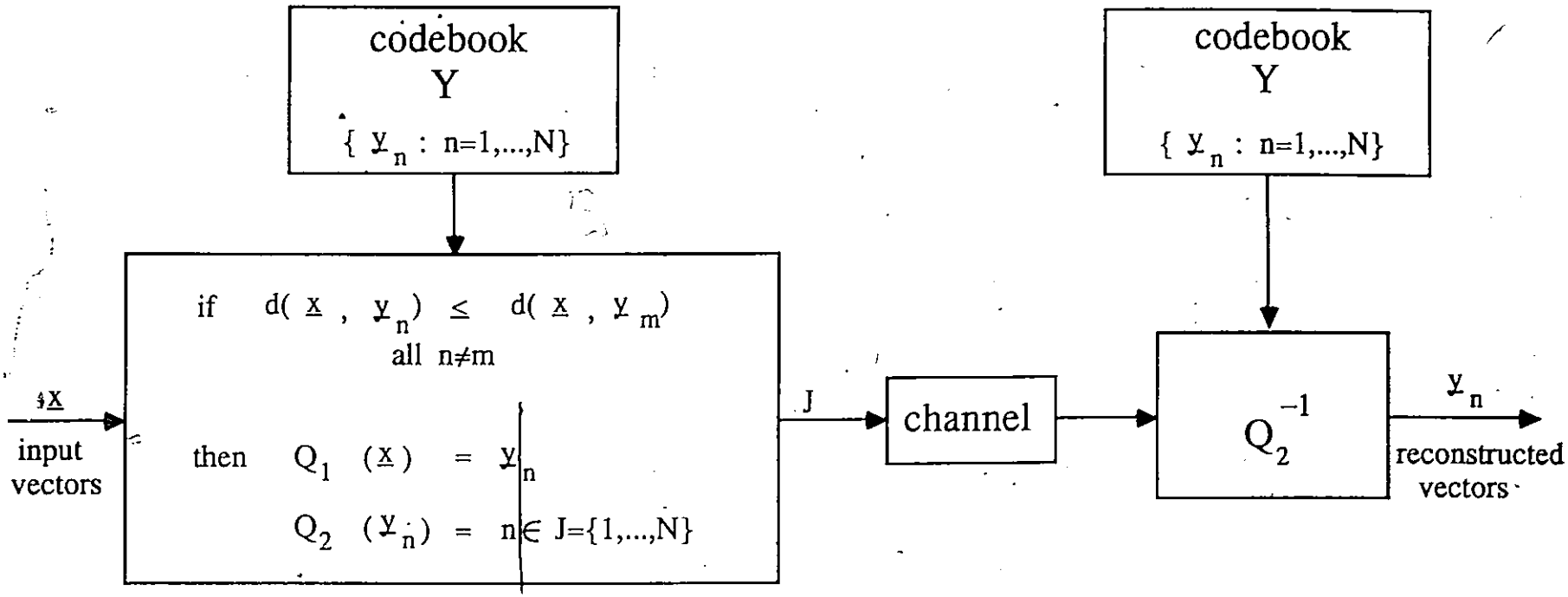


Fig. 2.1 The basic structure of vector quantization

with k for a given r. A VQ must search for all N codewords in order to select the one with minimum distortion to a given input vector. From (1),

$$N = 2^{kr} \qquad (2)$$

Thus, for a given rate r, a storage of $k(2^{kr})$ memory words is required and the number of computations per codeword or the number of computations per dimension grows exponentially with the dimension k of the input vector. For this practical implementation difficulty, the largest dimension of VQ (applies directly to the sampled speech waveform) is limited to k=8 with today's technology [24,25].

2.3 Performance parameters of a VQ

As discussed in section 2.2, the performance of a vector quantizer depends on how accurate the input vectors are represented by the codewords. It is clear that the codebook Y in the coder and the distortion measure $d(\underline{x},\underline{y})$ together determine the average distortion defined as

$$D(Q_1) = E(d(\underline{x},Q_1(\underline{x})))$$

where E is the expectation operator with respect to the probability density function of $\underline{x}$.

However, $D(Q_1)$ is often impossible to calculate because of the probability density function of $\underline{x}$ is not known. For instance, there are no generally accepted accurate probability density functions for real speech and image data. Therefore, in practice, $D(Q_1)$ is estimated by the sample average based on a long training sequence,

$X=(\underline{x}_0,\ldots,\underline{x}_{LT-1})$, and is defined as

$$D(Q_1) = \lim_{LT\to\infty} \frac{1}{LT}\left\{\sum_{n=0}^{LT-1} d(\underline{x}_n, Q_1(\underline{x}_n))\right\}$$

The sample average distortion will converge to the expected distortion, the expectation of $d(\underline{x},Q_1(\underline{x}))$, if the training sequence (the source) is stationary or ergodic. Ideally, one would like to design the codewords such that the expected distortion is minimized. However, for a source which is neither stationary nor ergodic and does not have a known probabilistic model, the resulting sample average is no longer the same as the expected value. Thus, in general, the expected distortion will not be minimized and the sample average distortion achieved by the designed codewords on the test sequence (other data sequences coming from the same source as the training sequence but not included in the training sequence) will be different (usually higher) than that achieved over the training sequence.

One method to deal with the problem stated above is to design the codewords based on a long training sequence and then examine the sample average distortions resulting from the training and the test sequences. If the sample average distortions are significantly different from one another, then redesign the codewords based on a longer training sequence. This approach is suggested in [1] and is experimentally supported in [6].

The codebook itself has two parameters namely the vector dimension k and the number of codewords N. They

jointly define the transmission rate r. It is important to note that either k or r contributes to the sample average distortion of a vector quantizer. Recall from section 2.2 equation (2),

$$N = 2^{kr}$$

A) At a given rate r, N increases with k. Thus, more codewords are used to quantize the input source as a higher vector dimension is used. As a result, the average distortion of a vector quantizer decreases with increasing k. This characteristic is detailed in [2,3] and is experimentally supported in [1,6,7].

B) At a given dimension k, N increases with r. Similar to A), the average distortion of a vector quantizer decreases with increasing r. This is also experimentally supported in [1,6,7].

Regardless of the physical size of the codebook, the codewords must be good enough to closely approximate the input source. The parameter that determines how accurate the input source is represented by the codewords is the average distortion. This parameter can also refer to the distortion measure as it identifies which codeword from the codebook is the best "match" to an input vector and it indicates how accurate the input vector has been "matched". The definition of "match" may be different for different sources. For the speech signal, we want the codewords to "sound like the source". In image coding, we want the codewords to "look like the source". As the choice of different distortion

measures for different sources is important, the choice of the physical size of the codebook is also important. A "good" distortion measure reflects the subjective judgement on a vector quantizer. The physical size of the codebook reflects the storage requirement and the computation complexity required for a VQ.

Ideally, one would like to design a VQ that performs as well as possible. However, for a given desired transmission rate, one has to increase the vector dimension to achieve lower average distortion. The exponential growth of the physical size of the codebook with the product of the vector dimension and transmission rate easily lead to an impractical VQ.

In short, for a source with unknown probability density function, the critical performance parameters of a VQ that have to be chosen with care are:

1) the length of the training sequence,

2) the vector dimension and the number of codewords,

3) the distortion measure.

2.4 Distortion measure

A useful distortion measure should be analytically tractable, real time computable and subjectively meaningful. In waveform coding, the most common distortion measure is the squared error measure (either weighted or unweighted). In voice coding, the commonly used distortion measures are the Itakura-Saito and the squared error measures. Although the squared error measure does not appear to be subjectively

meaningful for a low bit rate speech coding system, it is largely used due to its tractability and computability. As vector quantization is applicable in waveform and voice coding systems, the distortion measures for both of waveform and voice coding will be discussed in the following subsections.

2.4.1 Distortion measures for waveform coding

Let $\underline{x}$ be a vector to be reproduced by the vector $\underline{y}$. The weighted squared error distortion measure that assigns a non-negative distortion value between $\underline{x}$ and $\underline{y}$ is defined as

$$d_W(\underline{x},\underline{y}) = (\underline{x}-\underline{y})^T W(\underline{x}-\underline{y}) \quad (3)$$

where W is a symmetric positive definite matrix. When W is an identity matrix, the distortion measure reduces to the traditional squared error measure as

$$d_I(\underline{x},\underline{y}) = (\underline{x}-\underline{y})^T(\underline{x}-\underline{y}) \quad (4)$$

Another choice of the weighting matrix, W, is the inverse of the covariance matrix of the random vector $\underline{x}$. This covariance matrix, M, is defined as

$$M = E\{[\underline{x}-E(\underline{x})][\underline{x}-E(\underline{x})]^T\} \quad (5)$$

This distortion measure reduces to

$$d_M(\underline{x},\underline{y}) = (\underline{x}-\underline{y})^T M^{-1}(\underline{x}-\underline{y}) \quad (6)$$

the so called Mahalanobis distortion measure which is found to be useful in many pattern classification applications including the classification problems of Japanese phonemes [6].

Since M is symmetric and positive definite, so is M^{-1}. M^{-1} can be decomposed into $M^{-1}=L^TL$ where L is a lower triangular matrix. Thus,

$$\begin{aligned} d_M(\underline{x},\underline{y}) &= (\underline{x}-\underline{y})^T L^T L(\underline{x}-\underline{y}) \\ &= [L(\underline{x}-\underline{y})]^T[L(\underline{x}-\underline{y})] \\ &= (L\underline{x}-L\underline{y})^T(L\underline{x}-L\underline{y}) \end{aligned} \qquad (7)$$

Therefore, the minimum Mahalanobis distortion is achieved by minimizing the squared error distortion between the vectors transformed by L. This implies that the computation of $d_M(\underline{x},\underline{y})$ can be reduced by performing $d_I(L\underline{x},L\underline{y})$ in searching for the nearest neighbor and in codebook design. This computation reduction effect will be discussed in section 2.5.

2.4.2 Distortion measures for voice coding

The distortion measures that will be discussed are referred to as the so called Linear Prediction Distortion Measures. They are, generally, applied for the quantization of the parameters of a speech linear prediction model [27,28]. According to this model, the spectrum of a speech frame (typically 20 ms to 30 ms) is represented by the frequency response of an all pole spectral shaping filter given by

$$H(Z) = \frac{G}{A(Z)} \qquad (8)$$

where $A(Z) = \sum_{n=0}^{p} a_n Z^{-n}$ with $a_0 = 1$, and

$$G^2 = E_p = \sum_{n=0}^{p} a_n r(n) \qquad \text{with} \quad r(n)$$

the short term autocorrelation sequence of the speech frame $s(n)$, $0 \le n \le N_s-1$, given by

$$r(i) = \sum_{n=0}^{N_s-1-i} s(n)s(n+i) \qquad \text{for } i=0,\ldots,p$$

where $r(i)=r(-i)$

That is, if $S(f)$ is the amplitude spectrum of the speech frame $s(n)$, $0 \le n \le N_s-1$, then it is represented by the frequency response of $H(Z)$. The order p is typically ranges from 10 to 12 or from 12 to 14 if the speech signal is band-limited to 4 kHz or 5kHz, respectively.

2.4.2.1 Squared error distortion measure.

In practice, the gain term, G, from (8) is quantized and transmitted separately. However, the filter coefficients $\{a_n\}$ are likely to cause instability of the filter upon quantization. The reflection coefficients $\{k_n\}$, which are the intermediate quantities in solving for $\{a_n\}$ via Durbin's recursive algorithm, are selected as the transmission parameters [28]. The procedure of solving the $\{a_n\}$ via Durbin's recursive algorithm is the following:

initialize algorithm as

(i) $E_0 = r(0)$

calculate the k_n based on the previous coefficients

$$\text{(ii)} \quad k_n = -\Big(r(n) + \sum_{j=1}^{n-1} a_j^{(n-1)} r(n-j) \Big) / E_{n-1}$$

calculate the n^{th} coefficients for the next recursion

(iii) $a_n^{(n)} = k_n$

(iv) $a_j^{(n)} = a_j^{(n-1)} + k_n a_{n-j}^{(n-1)}$, $1 \le j \le n-1$

(v) $E_n = (1-(k_n)^2)E_{n-1}$

goto (ii) for the next recursion

By solving the equations (i) to (v) recursively, the final solution is given by

$$a_j = a_j^{(p)} \quad , \quad 1 \le j \le p$$

In addition, a study on the spectral sensitivity (the rate of change in dB in the amplitude spectrum of H(Z) with respect to the change in values of the corresponding reflection coefficients) shows that high or low spectral sensitivity occurs when $|k_n| \rightarrow 1$ or when $|k_n|$ is close to zero, respectively [29]. That is, small changes in those $|k_n| \rightarrow 1$ result in large changes in the amplitude spectrum. Therefore, linear quantization of the reflection coefficients is not optimal. An optimal quantization scheme proposed in [29] is a linear quantization on the log area ratios (g_n) defined as

$$g_n = \frac{1}{2L} \log \left(\frac{1+k_n}{1-k_n} \right) \quad , \quad 1 \le n \le p \qquad (9)$$

where L is an arbitrary constant. In this manner, the input vector $\underline{x} = (g_1, g_2, \ldots, g_p)^T$ will be encoded by another vector using (4), the squared error distortion measure [19].

2.4.2.2 Itakura-Saito distortion measure

Unlike the squared error distortion measure, the Itakura-Saito distortion measure is able to explore the spectral matching effect between two vectors. It is, therefore, believed to be a subjectively meaningful distortion measure [12].

According to this measure, the input vector (the parameters of H(Z)), is defined as $\underline{x}=(G,a_1,a_2,\ldots a_p)^T$. Let $\underline{x}$ be reproduced by an arbitrary H'(Z) with parameters defined as $\underline{y}=(G',a'_1,a'_2,\ldots,a'_p)^T$, then the Itakura-Saito distortion [4,10,12] between $\underline{x}$ and $\underline{y}$ is given by

$$d_{IS}(\underline{x},\underline{y}) = \frac{\underline{a}^T R(S)\underline{a}}{G'^2} + \ln G'^2 - \ln G^2 - 1 \qquad (10)$$

where $\underline{a}=(1,a'_1,a'_2,\ldots,a'_p)^T$ and R(S) is the p+1 by p+1 autocorrelation matrix , $(r_s(i-j),\ i=0,1,2,\ldots,p;\ j=0,1,2,\ldots,p)$, with $r_s(i)$ the standard short term autocorrelation sequence for the truncated (windowed) speech sample s(n), $0\le n\le N_s-1$, given by

$$r_s(i) = \sum_{n=0}^{N_s-1-i} s(n)s(n+i) \qquad \text{for } i=0,1,2,\ldots,p \qquad (11)$$

where $r_s(i)=r_s(-i)$

In [4,10,12], it is shown that

$$\underline{a}^T R(S)\underline{a} = r_{a'}(0)r_s(0) + 2\sum_{i=1}^{p} r_{a'}(i)r_s(i) \qquad (12)$$

$$\text{where } r_{a'}(i) = \sum_{m=0}^{p-i} a'_m a'_{m+i} \ ;\ r_{a'}(i) = r_{a'}(-i) \qquad (13)$$

A useful property of the Itakura-Saito distortion measure is the "triangular equality" [4,12]. Rewrite (10) as

$$d_{IS}(\underline{x},\underline{y}) = \ln[\underline{a}^T R(S)\underline{a}] - \ln G^2 + \frac{\underline{a}^T R(S)\underline{a}}{G'^2} - \ln[\underline{a}^T R(S)\underline{a}] + \ln G'^2 - 1$$

$$- \left\{ \ln\left(\frac{\underline{a}^T R(S)\underline{a}}{G^2} \right) \right\} + \left\{ \frac{\underline{a}^T R(S)\underline{a}}{G'^2} - \ln\left(\frac{\underline{a}^T R(S)\underline{a}}{G'^2} \right) - 1 \right\}$$

$$= d(\underline{x},\underline{a}) + d(\underline{a}^T R(S)\underline{a}\ ,\ G'^2) \qquad (14)$$

where $d(\underline{x},\underline{a})$ is the gain-optimized Itakura-Saito distortion measure [12]. It is the Itakura-Saito distortion when $\underline{x}$ is encoded by $\underline{a}$ and a gain term of $(\underline{a}^T R(S)\underline{a})^{1/2}$; and $d(\underline{a}^T R(S)\underline{a}, G'^2)$ is the Itakura-Saito distortion when $(\underline{a}^T R(S)\underline{a})^{1/2}$ is quantized by G'. This separation of $d_{IS}(\underline{x},\underline{y})$ is called the triangular equality.

Because of the additive property results from (14), $d_{IS}(\underline{x},\underline{y})$ can be minimized by separately minimizing $d(\underline{x},\underline{a})$ and $d(\underline{a}^T R(S)\underline{a}\ ,\ G'^2)$.

Step 1) Choose a model (without the gain term) $\underline{a}_* = (1, a_{*1}, \ldots, a_{*p})^T$ such that

$$d(\underline{x},\underline{a}_*) \leq d(\underline{x},\underline{a})$$

with some tie-breaking rule.

Step 2) Subject to the model $\underline{a}_*$ chosen in step 1), choose a gain term G_* such that

$$d(\underline{a}_*^T R(S)\underline{a}_*, G_*^2) \leq d(\underline{a}_*^T R(S)\underline{a}_*, G'^2)$$

with some tie-breaking rule.

Step 1) is a vector quantization that does not depend on the gain term. Step 2) is a scalar quantization and

$d(\underline{a}_*^T R(S)\underline{a}_*, G_*^2)=0$ iff $G_*=(\underline{a}_*^T R(S)\underline{a}_*)^{1/2}$. Therefore, (14) can be written as

$$d_{IS}(\underline{x};\underline{a}_*,G_*) = d(\underline{x},\underline{a}_*) + d(\underline{a}_*^T R(S)\underline{a}_*, G_*^2) \qquad (15)$$

which leads to the development of another vector quantizer called Gain Shape Vector Quantizer (GSVQ) [4,10].

Having completed the discussion of these distortion measures, it is clear that voice coding system does require complicated computation. In the next section, the applications of vector quantization in both waveform and voice coding systems are discussed.

2.5 Codebook design

Recall that a desired VQ for a given codebook size should be able to minimize the average distortion between the source and the quantized output (represented by the codewords). Thus, a well performing VQ implies a set of good codewords and subjectively a good distortion measure. Define a set of partitions (a partition is a "bounded" multi-dimensional region of dimension equal to the vector dimension), $(P_n, n=1,\ldots,N)$. Each partition relates to each codeword $(\underline{y}_n)$, $1\le n\le N$ as

$$\underline{x} \in P_n \qquad \text{if} \quad d(\underline{x},\underline{y}_n) \le d(\underline{x},\underline{y}_m) \quad \text{for all } n \ne m$$

Therefore, from the step 1) in section 2.2,

$$Q_1(\underline{x}) = \underline{y}_n \quad \text{if } \underline{x} \in P_n$$

The average distortion is, therefore, defined as

$$D(Q_1) = E(d(\underline{x},Q_1(\underline{x})))$$

$$= E\left\{ \sum_{n=1}^{N} \left[d(\underline{x},\underline{y}_n) \middle| \underline{x} \in P_n \right] P_r(\underline{x} \in P_n) \right\}$$

$$= \sum_{n=1}^{N} E\left\{\left[d(\underline{x},\underline{y}_n)\,\middle|\, \underline{x} \in P_n\right] P_r(\underline{x} \in P_n)\right\}$$

As discussed in section 2.3, $D(Q_1)$ is estimated by the sample average from the training sequence, $X=(\underline{x}_0,\ldots,\underline{x}_{LT-1})$. Thus,

$$D(Q_1) = \frac{1}{LT}\sum_{n=0}^{LT-1} d(\underline{x}_n, Q_1(\underline{x}_n))$$

$$= \frac{1}{LT}\sum_{n=0}^{LT-1}\sum_{m=1}^{N}\left[d(\underline{x}_n,\underline{y}_m)\,\middle|\, \underline{x}_n \in P_m\right] P_r(\underline{x}_n \in P_m)$$

where $P_r(\underline{x}_n \in P_m)$ is the discrete probability which is either 1 or 0.

A VQ is optimal if it minimizes $D(Q_1)$, that is, a VQ with Q_1^* is optimal if all other VQ with Q_1 result in $D(Q_1^*)\leq D(Q_1)$. There are two necessary conditions for an optimal VQ [5,14].

Condition 1) $Q_1(\underline{x})=\underline{y}_n$ if $\underline{x} \in P_n$. Note that this is exactly the same as the step 1) in section 2.2.

Condition 2) $\underline{y}_n$ minimizes the sample average distortion in partition P_n, that is

$$D_n(Q_1) = \frac{1}{N_n}\sum_{m=1}^{N_n} d(\underline{x}_m,\underline{y}_n) \qquad (16)$$

is minimized by $\underline{y}_n$; where N_n=number of vectors $\underline{x}$ in P_n, and $\underline{x}_m \in P_n$ for all $1\leq m\leq N_n$. The codeword $\underline{y}_n$ is called the centroid of P_n or $\underline{y}_n=\text{cent}(P_n)$.

Having discussed different distortion measures available in speech coding, it is possible to discuss in

detail how to conduct the search for the nearest neighbors of the input vectors (condition 1 of an optimal VQ) and the computation of the centroids (condition 2 of an optimal VQ).

2.5.1 Nearest neighbor calculation

From condition 1) of an optimal VQ, it is obvious that the computation involved in the nearest neighbor calculation is dependent on the distortion measure.

2.5.1.1 Weighted squared error distortion measure

From (3), the general form of a weighted squared error distortion between a k-dimensional input vector $\underline{x}$ and the k-dimensional codewords $(\underline{y}_n)$, $1 \le n \le N$ is given by

$$d_W(\underline{x}, \underline{y}_n) = (\underline{x}-\underline{y}_n)^T W(\underline{x}-\underline{y}_n) \quad (17)$$

$$= \underline{x}^T W \underline{x} - 2\underline{x}^T W \underline{y}_n + \underline{y}_n^T W \underline{y}_n \quad (18)$$

Since the first term is independent of $\underline{y}_n$, it can be ignored in the nearest neighbor calculation. If the coder is designed to store $-2W\underline{y}_n$ (k-dimensional vector) and $\underline{y}_n^T W \underline{y}_n$ (scalar) instead of $\underline{y}_n$ for $1 \le n \le N$, then the required computation of (18) is less than that of (17).

Let add. denotes addition(s),

mult. denotes multiplication(s),

(add.+mult.) denotes additions and multiplications

sub. denotes subtraction(s), and

the computation of $W\underline{z}$ be Wc= k[k mult.+(k-1) add.],

where $\underline{z}$ is an arbitrary k-dimensional vector.

1) The required computation involved in (17) is

k sub. + k mult. + (k-1) add. + Wc (add.+mult.).

2) The required computation involved in (18) is

k mult. + (k-1) add. + 1 add.

Thus, the saving in the required computation in (18) compared to (17) is

k sub. - 1 add. + Wc (add.+mult.)

It follows directly that the saving in computation for the nearest neighbor calculation for N codewords is

N [k sub. - 1 add. + Wc (add.+mult.)]

The price to pay for this saving in the number of computations is the increase in storage of N scalar terms. Therefore, the computation reduction achieved by (18) is a trade off between computation and storage.

As pointed out in section 2.4, the computation for the Mahalanobis distortion in (6) can be reduced by performing the squared error distortion measure in (7). In order to compute the Mahalanobis distortion defined in (7), the coder has to be designed to store the transformed vectors $L\underline{y}_n$ instead of $\underline{y}_n$ for $1 \le n \le N$.

Let the computation of $M\underline{z}$ be Mc (add.+mult.), and the computation of $L\underline{z}$ be Lc (add.+mult.), where $\underline{z}$ is an arbitrary k-dimensional vector.

1) The required computation in calculating $d_M(\underline{x}, \underline{y}_n)$ through (6) for all n is

N [k sub. + k mult. + (k-1) add. + Mc (add.+mult.)].

2) The required computation in calculating $d_M(\underline{x}, \underline{y}_n)$ through (7) for all n is

N [k sub. + k mult. + (k-1) add.].

3) The precomputation of the input vector $\underline{x}$ before $d_M(\underline{x},\underline{y}_n)$ through (7) for all n is

Lc (add.+mult.).

Since $M=L^TL$, assume Mc=Lc. Thus the saving in computation for the nearest neighbor calculation achieved by (7) is

NMc-Lc=Mc(N-1) (add.+mult.)

In addition, no extra storage is required for this computation reduction.

2.5.1.2 Itakura-Saito distortion measure

From (10) , the Itakura-Saito distortion between the input model vector $\underline{x}=(G,a_1,a_2,\ldots a_p)^T$ and the codewords $(\underline{y}_n=(G'_n,a'_1(n),a'_2(n),\ldots a'_p(n))^T)$, $1\le n\le N$ is given by

$$d_{IS}(\underline{x},\underline{y}_n) = \frac{[\underline{a}(n)]^T R(S)\underline{a}(n)}{G'^2_n} + \ln G'^2_n - \ln G^2 - 1 \qquad (19)$$

where $\underline{a}(n)=(1,a'_1(n),a'_2(n),\ldots a'_p(n))^T$

Since the last two terms are both independent of $\underline{y}_n$, they can be ignored in the nearest neighbor calculation. Define $d'_{IS}(\underline{x},\underline{y}_n)$ as

$$d'_{IS}(\underline{x},\underline{y}_n) = \frac{[\underline{a}(n)]^T R(S)\underline{a}(n)}{G'^2_n} + \ln G'^2_n \qquad (20)$$

substitude (12) into (20),

$$d'_{IS}(\underline{x},\underline{y}_n) = \left[r_{a'(n)}(0)r_s(0) + 2\sum_{m=1}^{p} r_{a'(n)}(m)r_s(m)\right] \Big/ G'^2_n + \ln G'^2_n \qquad (21)$$

where $r_{a'(n)}(k)=\sum_{m=0}^{p-k} a'_m(n)a'_{m+k}(n)$

Thus, for computation purposes, each codeword should be stored as $(\mu_n=(\ln G'_n{}^2,\ r_{a'(n)}(0)/G'_n{}^2,\ 2r_{a'(n)}(1)/G'_n{}^2,\ldots,\ 2r_{a'(n)}(p)/G'_n{}^2)\ )$, $1\le n\le N$.

2.5.2 Centroid calculation

Similar to the nearest neighbor calculation, the centroid computation depends on the distortion measure.

2.5.2.1 Weighted squared error distortion measure

In calculating the centroid, we want to minimize (16) for all $1\le n\le N$. For a distortion measure of the form (3), (16) is minimized with the centroid shown in [5] as

$$\mathrm{cent}(P_n) = \left[\sum_{m=1}^{N_n} W\right]^{-1} \sum_{m=1}^{N_n} W\underline{x}_m \qquad (22)$$

where N_n and $\underline{x}_m$ is the same as in (16).

As discussed in section 2.4, $W=L^TL$ and is independent of $\underline{x}$, thus (22) becomes

$$\begin{aligned}\mathrm{cent}(P_n) &= \frac{1}{N_n}(L^TL)^{-1}\sum_{m=1}^{N_n}(L^TL)\underline{x}_m \\ &= \frac{1}{N_n}L^{-1}(L^T)^{-1}L^TL\sum_{m=1}^{N_n}\underline{x}_m \\ &= \frac{1}{N_n}\sum_{m=1}^{N_n}\underline{x}_m \qquad (23)\end{aligned}$$

Note that $\mathrm{cent}(P_n)$ is independent of W.

2.5.2.2 Itakura-Saito distortion measure

Let $(S_m(f):\ 1\le m\le N_n)$ be the energy density spectra of the speech frames which are represented by a set of spectral shaping filters $(H_m(z):\ 1\le m\le N_n)$, each with a parameter

vector or codeword: $\underline{x}_m=(G'_m,a'_{m;1},a'_{m;2},\ldots,a'_{m;p})^T$. In [4,12], it is shown that the required cent(Pn) is the parameter vector $\underline{x}^{\sim}$ corresponding to

$$S^{\sim}(f) = \frac{1}{N_n}\sum_{m=1}^{N_n} S_m(f) \qquad (24)$$

where $S^{\sim}(f)$ is the arithmetic mean of $\{S_m(f): 1\leq m\leq N_n\}$. Thus, the calculation of cent(P_n) is the linear prediction problem of $S^{\sim}(f)$.

Define $\{r_s^{(m)}(n): 1\leq m\leq N_n;\ 0\leq n\leq p\}$ the short term autocorrelation sequences for the speech frames and $\{S_m(Z): 1\leq m\leq N_n\}$ the Z - Transform of the speech frames. Thus,

$$\begin{aligned} S_m(f) &= S_m(Z)S_m(1/Z) \\ &= Z(r_s^{(m)}(n)) \end{aligned} \qquad (25)$$

Since the Z - Transform is linear, $S^{\sim}(f)$ can be defined as

$$\begin{aligned} S^{\sim}(f) &= \frac{1}{N_n}\sum_{m=1}^{N_n} S_m(f) \\ &= \frac{1}{N_n}\sum_{m=1}^{N_n} Z(r_s^{(m)}(n)) \\ &= Z\left\{\frac{1}{N_n}\sum_{m=1}^{N_n} r_s^{(m)}(n)\right\} \\ &= Z(r_s^{\sim}(n)) \end{aligned} \qquad (26)$$

From (25) and (26), it is obvious that $\underline{x}^{\sim}$ can be calculated by means of the autocorrelation method [28] based upon the arithmetic mean of the autocorrelation sequences for the speech frames.

2.5.3 The LBG algorithm

One method for codebook design, subject to the optimal VQ criteria, is an iterative clustering algorithm called the LBG algorithm or the generalized Lloyd algorithm [5]. It works with a general distortion measure, a long vector dimension and a "general source". A "general source " is a source that either has known probability distribution or that has an unknown probabilistic model. For a source that has unknown probabilistic model, a very long training sequence generated by the source is used as an approximation of the "true" but unknown probability distribution of the source.

The basic of the LBG algorithm is the following:

Step 0) Given N, the number of codewords required ; k , the dimension of each codeword and a training sequence

$$X=\{\underline{x}_i: \underline{x}_i=(x_{1i},\ldots,x_{ki}),\ 0\le i\le LT-1\}$$

Initialize a N codewords codebook $Y_0=\{\underline{y}_{0j}: 1\le j\le N\}$, a distortion threshold $D_t\ge 0$, an iteration index $n=0$ and assume a distortion $D_{-1}=\infty$.

Step 1) Find the set of partitions $\{P_{nj}, j=1,\ldots,N\}$ based on Y_n and X, then find the reproduction vectors $\{\underline{y}_{nj}\}$ according to condition 1) of an optimal VQ. Compute the sample average distortion

$$D_n = \frac{1}{LT}\sum_{i=0}^{LT-1} d(\underline{x}_i, Q_1(\underline{x}_i))$$

Step 2) If $(D_{n-1}-D_n)/D_n\le D_t$, then halt with a final codebook Y_n. Otherwise go to step 3).

Step 3) Find the optimal codebook according to condition 2) of an optimal VQ.

$$Y_n=\{y_{nj}=cent(P_{nj}):\ j=1,\ldots,N\}$$

increment n and go to step 1).

Since the design of the final codebook requires an initial codebook Y_0 and a training sequence, the final codebook is optimal with respect to the training sequence and Y_0 or the final codebook is locally optimum [5].

Therefore, it is often suggested that for an arbitrary source, different Y_0 should be tried in designing the codebook and the final codebook that results with the minimum distortion is then chosen to be the desired codebook.

One codebook initialization method is found to be useful and is extensively used for speech source is the splitting technique [1,4,5,6,7,15,18]. This method starts with the centroid of the training sequence as a single codeword. This codeword is then split into two codewords to create the Y_0 for the LBG algorithm. After the optimal codebook (2 codewords) is obtained, each of these two final codewords is then split into two initial codewords as the Y_0 for the LBG algorithm. As a result, one obtains a codebook of size 1,2,4,...,N codewords sequentially. In addition, according to the results reported in [6], the splitting initialization technique is expected to be a good codebook initialization method (for a speech source) in the sense

that the final codebook is optimal with respect to the training sequence.

The LBG algorithm together with the splitting initialization technique is the following (a subroutine program is given in Appendix C):

Step 0) Given N , the desired number of codewords ; k , the dimension of each codeword and a training sequence

$$X=\{\underline{x}_i: \underline{x}_i=(x_{1i},\ldots,x_{ki}),\ 0\leq i\leq LT-1\}.$$

Compute the centroid of X, $\underline{y}_0$. Set m=1.

Step 1) Given a m-level codebook $Y=\{\underline{y}_i: 1\leq i\leq m\}$ split[1] each of the codeword $\underline{y}_i$ into $\underline{y}_i$ and $\underline{y}_i+\underline{b}$ where $\underline{b}$ is a fixed perturbation vector[2]. Set up the initial codebook $Y_0=\{\underline{y}_i: 1\leq i\leq 2m\}$ with 2m codewords.

Step 2) Find the optimal 2m-level codebook , Y_f, via the LBG algorithm.

Step 3) If 2m<N , replace Y by Y_f, m by 2m and go to step 1). Otherwise, halt with Y_f as the desired codebook.

[1] Y is the optimal m-level codebook with minimum average distortion (arising when X is reproduced by Y). The initial 2m-level codebook Y_0 which includes a copy of Y ensures the average distortion arising when X is reproduced by Y_0, will not be higher than that resulting from Y. Thus the average distortion arising when X is reproduced by Y_f is guaranteed to be lower than that resulting from Y.

[2] The purpose of the vector $\underline{b}$ is to create two "closely packed" vectors in the k-dimensional Euclidean vector space. The nature of $\underline{b}$ is arbitrary in that it can be a vector of different vector components, a vector of equal vector components or even a scalar quantity.

2.6 The design procedure of the VQ

Having discussed the structure, the performance parameters and the codebook design of VQ, the design of a particular VQ can be considered. In practical systems, speech coders are usually classified according to their transmission bit rates. Any speech coder, therefore, must provide the information on its output speech quality as well as on its transmission bit rate.

A general approach in speech coder development is to enhance the output speech quality subject to a particular transmission bit rate. Therefore, it is of interest in designing a VQ subject to a desired transmission bit rate.

The procedure in designing a VQ subject to a given transmission bit rate can be formulated as follows:

Step 1) Select a suitable distortion measure for the input vectors. For instance, the Itakura-Saito distortion measure in linear predictive coding. An example is given in section 3.2.3.

Step 2) Select the largest possible vector dimension that can be handled through the computer (non real-time) simulation and the required length of the training sequence. In linear predictive coding, the vector dimension is usually chosen to be the number of the model parameters. Thus, step 2) does not, usually, apply in linear predictive coding. An example is given in section 3.2.3.

Step 3) Select the codebook size (the number of codewords), subject to the selected vector dimension and the required transmission bit rate. An example is given in section 3.2.3.

Step 4) Select a training sequence based upon the selected vector dimension. Due to the exponential growth of the number of codewords with the vector dimension, the length of the training sequence increases exponentially with the vector dimension. An example is given in section 3.2.4.

Step 5) Design the codebook using the LBG algorithm.

Step 6) Test the designed codebook on an arbitrary test sequence which is also from the same source as the training sequence. An example is given in section 3.2.4.

Step 7) If the result on the test sequence does not match the result obtained from the training sequence, redesign the codebook base on a longer training sequence.

The design procedure outlined above provides a systematic approach in designing a VQ. However, both the step 2) and the step 4) depend heavily on the statistical behaviour of the training sequence (the source) and the available computer facilities. The statistical behaviour of the source determines the required length of the training sequence. The computer facilities determine the length of

the vector dimension and the length of the training sequence that can be handled through the non real-time simulation.

As an example for the application of this design procedure, an 8 kb/s VPCM will be designed following this procedure in chapter 3 (section 3.2.3, 3.2.4).

2.7 Conclusion

In this chapter, the basic method of full search vector quantization is given. Out of the materials covered, the LBG algorithm is of special importance. This algorithm is the basic tool for the codebook design and is applicable in designing the codebooks of mostly all the different types of vector quantizers [4,6,7,8,9,15,18,19,20,22], except an extension is required for the FSVQ [11] and the GSVQ [10].

Finally, all of the materials covered in this chapter will be referred to and will not be repeated in the following chapters. Therefore, the reader may find that it is necessary to refer back to this chapter occasionally during the reading of the following chapters.

CHAPTER 3

DEVELOPMENT OF THE MEDIUM-BAND SPEECH CODER

3.1 Introduction

As discussed in chapter 1, the desired speech coder is based upon VPCM. Hence, three different VPCM-based systems are developed. Each of these extensions of VPCM is designed to improve the performance of VPCM while keeping to a minimum, the increases in the computation complexity and in the transmission bit rate. As a result, every realization of the different VPCM-based systems is a compromise between the computation complexity, the transmission bit rate and the performance.

The first extension of VPCM (sections 3.4, 3.5, 3.6) is a technique that reduces redundancy prior to vector quantization. The redundancy reduction makes use of the pitch period to pitch period and the cycle to cycle dependencies of the speech waveform. These dependencies are not effectively addressed by VPCM. However, this VPCM-based system is very complicated. Its final development is emphasized on simplifing the redundancy reducing technique.

The second extension of VPCM (sections 3.7, 3.8) is an adaptive technique that makes use of an adaptive vector quantizer (codebook) for the speech waveform. This adaptive technique is designed generally in combination with the first extension of VPCM.

The third extension of VPCM (sections 3.7, 3.8) is a combination of the redundancy reducing and adaptive techniques. Each of these variations of VPCM including VPCM will be discussed in detail throughout this chapter. In section 3.9, a comparison between the different extensions of VPCM based upon their output speech quality and transmission bit rate is given. After the comparison, the third extension of VPCM is concluded to be the desired speech coder in section 3.10.

3.2 Vector pulse code modulation (VPCM)

The simplest speech coding scheme that requires the highest transmission bit rate (around 96 kb/s) is the linear pulse code modulation (PCM). The encoding process of PCM is basically the conversion of the analog (continuous-time, continuous-amplitude) speech waveform into a digital (discrete-time, discrete-amplitude) speech waveform. The analog to digital conversion consists of three parts: sampling, quantization and coding. Sampling converts a continuous-time signal into a discrete-time signal according to the Sampling Theorem. Quantization converts a continuous-amplitude signal into a discrete-amplitude signal based on a finite set of discrete amplitude. The finite set of discrete amplitude (codebook) is time-invariant and must be defined before the A/D conversion of the speech waveform. The transmission parameter is the index of the discrete amplitude (codeword) from the codebook. The decoding process of PCM is the digital to analog conversion (D/A): the

recovering of the codewords from their indexes followed by low-pass filtering.

The difference between VPCM and PCM is the different nature of the quantization process. In PCM, each sample from the sampled speech waveform is independently quantized. In VPCM, a block of consecutive samples from the sampled speech waveform is quantized through vector quantization. Each vector is also independently quantized by the vector quantizer. Because of this similarity, VPCM is considered as a direct generalization of the conventional linear PCM.

The vector quantizer in VPCM can be any kind of memoryless vector quantizer. However, as discussed in chapter 1, all the memoryless vector quantizers except the full search vector quantizer (VQ) are suboptimal. Therefore, only the VPCM with full search vector quantizer is of interest. "VPCM" will stand for vector pulse code modulation together with full search vector quantizer and "VQ" will stand for the encoding process of the full search vector quantizer throughout the rest of this and the following chapters. A block diagram of VPCM is shown in Fig. 3.1.

3.2.1 Transmission bit rate of VPCM

Recall that (chapter 2), the transmission rate of a N-codeword k-dimensional full search vector quantizer is

$$r = \frac{\log_2 N}{k} \quad \text{bits/dimension}$$

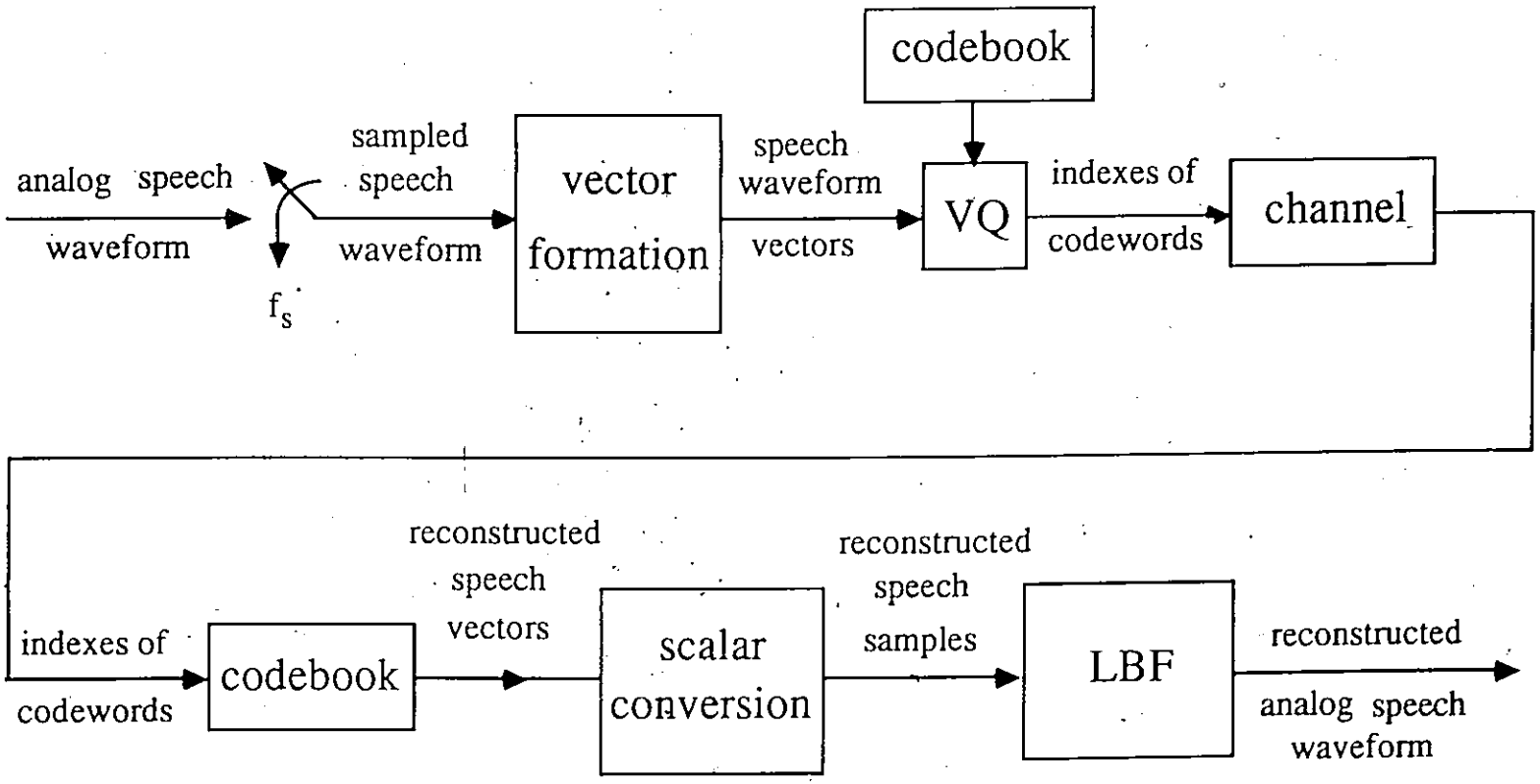


Fig. 3.1 Block diagram of VPCM

VQ : the encoding process of the full search vector quantizer
LBF : lowpass filter
f_s : sampling frequency

In VPCM, the vector dimension is equal to the number of speech samples per vector. Thus, the transmission rate of VPCM is

$$r = \frac{\log_2 N}{k} \quad \text{bits/sample}$$

With a sampling rate of f_s samples per second, the transmission bit rate of VPCM is

$$R = \frac{\log_2 N}{k} \cdot f_s \quad \text{bits/sec} \qquad (27)$$

3.2.2 Performance of VPCM

As described in chapter 1, an infinite-dimensional VPCM can, theoretically, achieve the distortion-rate function. However, only the finite-dimensional VPCM is practically useful. Additionally, the computation complexity of VPCM, discussed in chapter 2 (section 2.2), virtually limits the dimension of VPCM to $k \leq 8$. Therefore, only the performance of VPCM with short (≤ 8) vector dimension is of interest here.

A detailed study on VPCM from Stanford University [6] implies that for a VPCM of fixed transmission rate (bits/sample) and of dimension k, the SNR improvement results from using a VPCM of dimension k+1 is decreasing as k is increasing. A similar result can also be obtained from an independent study on VPCM in Germany [15]. It should be emphasized that the relation between the performance of VPCM and its vector dimension has never been reported in the literature. Therefore, it is necessary to confirm it in the following.

Some of the results of the VPCM in [6] are shown in Table 3.1. According to the SNR_{out} at 1 bit/sample, an increase of 3.2 dB in SNR was obtained when k was increased from one to two and an increase of less than 0.5 dB in SNR was obtained per unit increase in k starting from k=5. Therefore, a vector dimension higher than eight (k>8) at 1 bit/sample may not be desirable as the required computation complexity and memory storage grows exponentially with k and the increase in SNR resulting from higher vector dimension is small.

3.2.3 Selection of VPCM parameters

Usually, the transmission bit rate of a medium-band speech coding scheme ranges from 8 kb/s to 16 kb/s and a commonly used sampling frequency, f_s, is 8 kHz. Although a sampling frequency of 6.5 kHz is also good for North American English, it is not good enough for some other languages such as German and French. In addition, lower sample to sample correlation results from the 6.5 kHz sampling frequency as compared to the 8 kHz sampling frequency. Therefore, a 8 kHz sampling frequency is selected as it is generally accepted and it contributes to better output speech quality than the 6.5 kHz sampling frequency [24]. In addition to the effect of higher sampling rate on the performance, a SNR of 13.6 dB is achieved by an 8-dimensional VPCM (at 1 bit/sample) using a sampling frequency of 8 kHz [15]. Although it is difficult (perhaps not possible) to compare the results obtained from [6] and

	1 bit/sample				2 bits/sample			
k	SNR_{in} (dB)	SNR_d (dB)	SNR_{out} (dB)	SNR_d (dB)	SNR_{in} (dB)	SNR_d (dB)	SNR_{out} (dB)	SNR_d (dB)
1	2.01		2.05		6.50		6.41	
2	5.24	3.23	5.25	3.20	9.98	3.48	9.72	3.31
3	6.09	0.85	6.01	0.76	12.13	2.15	11.85	2.13
4	7.09	1.00	6.99	0.98	13.48	1.35	12.70	0.85
5	7.88	0.79	7.57	0.58				
6	8.53	0.65	8.05	0.48				
7	9.08	0.55	8.41	0.36				
8	9.74	0.66	8.82	0.41				

Table 3.1 Performance of VPCM at 1 bit/sample and 2 bits/sample. After [6].

The VPCM was designed based on a training sequence (6.5 kHz sampled real speech containing five speakers) of 640000 samples and was tested on a test sequence (6.5 kHz sampled real speech of a single speaker who was not involved for the training sequence) of 78600 samples. The design algorithm was the LBG algorithm based on the unweighted squared error distortion measure together with the splitting initialization and a convergence threshold of 0.005. SNR_{in} is the SNR obtained from the training sequence. SNR_{out} is the SNR obtained outside the training sequence (from the test sequence). SNR_d is the increase in SNR per unit increase in vector dimension, k, defined as

SNR_d(dB) at dimension k = SNR (dB) at dimension k - SNR (dB) at dimension k-1

or $SNR_d(dB) @ k = SNR (dB) @ k - SNR (dB) @ k-1$

As an example, the SNR_d of SNR_{in} at k=8 is

$$SNR_d(dB) @ 8 = SNR_{in} (dB) @ 8 - SNR_{in} (dB) @ 7$$
$$= 9.74 - 9.08$$
$$= 0.66 \text{ dB}$$

[15] because of the different nature of the training sequences and test sequences used in each case, these results certainly concur on the advantage of using the 8 kHz sampling frequency over the 6.5 kHz sampling frequency. Approximately a 4 dB increase in SNR is achievable by using the higher sampling rate.

The disadvantage of using the 8 kHz sampling frequency as compared to the 6.5 kHz sampling frequency is the resulting increase in the transmission bit rate. However, according to [6] and [15], the extra 1.5 kb/s transmission bit rate is acceptable because of the corresponding SNR improvement.

Based upon an 8 kHz sampling frequency, the admissible transmission rate for a medium-band VPCM is between 1 bit/sample and 2 bits/sample. Accounting for the extra bit rate for word synchronization and the side information required by the further development (the redundancy reducing, and the adaptive techniques), a VPCM of 1 bit/sample is selected. Also, according to (27), the corresponding transmission bit rate, which does not include the bit rate for word synchronization, of the VPCM is 8 kb/s. Note that the 8 kb/s transmission bit rate is the minimum transmission bit rate of all the different extensions of VPCM. Furthermore, based on the discussion on the effect of the vector dimension in section 3.2.2, and the computation complexity requirement discussed in chapter 2 (section 2.2), a vector dimension, k, of 8 has been selected. The increase

in computation required for k>8 is not proportional to the corresponding SNR improvement; thus the highest vector dimension for VPCM, which is implementable, is k=8 [24, 25]. Finally, according to (2), the number of codewords, N, of the 1 bit/sample VPCM is 2^8=256.

The last parameter, which is necessary to be determined for the nearest neighbor selection, is the distortion measure. As discussed in chapter 2 (section 2.4), the most popular distortion measures for waveform coding systems are the unweighted squared error distortion measure and the Mahalanobis distortion measure. However, the Mahalanobis distortion measure does not appear to be better than the unweighted squared error distortion measure in VPCM [6]. According to [6], the subjective output speech quality of the unweighted squared error distortion results is almost identical to the corresponding Mahalanobis distortion results. Therefore, the unweighted squared error distortion measure is chosen for the VPCM.

As a summary, the parameters of the selected VPCM are the following:

Distortion measure	= unweighted squared error distortion measure
Sampling frequency	= 8 kHz
Number of codewords	= 256
Vector dimension	= 8
Transmission rate	= 1 bit/sample
Transmission bit rate	= 8 kb/s

3.2.4 Simulation results of the selected VPCM

Based upon the parameters chosen from the previous section, the 256-codewords 8-dimensional VPCM was simulated in the VAX 11/750 mini-computer. Real speech, from four different speakers, of about 33 seconds (264080 samples) long was used as the training sequence for codebook design. Similarly, the test sequence was another real speech sequence, recorded from a single speaker, of about 11 seconds (87280 samples) long. The real speech[3] (the training sequence and the test sequence) was sampled at 8 kHz and quantized to 16-bit.

A codebook of 256 codewords was designed for the 8-dimensional VPCM using the LBG algorithm with the splitting initialization and a convergence threshold of 0.005 which was found to be useful [5,6,7]. The simulated performances of the VPCM are the following:

The SNR of the VPCM obtained inside the training sequence = 11.60 dB

The SNR of the VPCM obtained inside the test sequence = 12.24 dB

Note that the performance of the selected VPCM inside the test sequence is quite close to that obtained inside the

[3]The real speech data was borrowed from Dr. Melvyn J. Hunt, a research officer in National Research Council. This speech data was recorded from five different speakers labelled as AA, CM, DS, GH AND JT. The speech data of each speaker was approximately 100 seconds.

training sequence. It is, therefore, assumed that the 33 seconds training sequence is long enough for the codebook design, according to the discussion on the average distortion $D(Q_1)$ in section 2.3.

Another observation that can be made from the simulation results is that the VPCM results in higher SNR than that obtained in [6]. Thus, the advantage of using an 8 kHz sampling frequency instead of the 6.5 kHz sampling frequency is, once again, confirmed by the simulation results.

3.2.5 Comments on VPCM

As discussed before, the performance of VPCM, though significantly better than that of PCM, is not good enough compared to some other scalar speech compression techniques. A SNR of 12.24 dB of the 8 kb/s VPCM is not good enough as a generally accepted medium-band speech coding system [15]. However, VPCM is still, potentially, a promising medium-band speech coding scheme since it achieves a SNR of about 12 dB without fully utilizing the redundant information in the speech waveform. As described in chapter 1, a set of properly designed codewords only allows VPCM to make effective use of the non-uniform probability density function of the amplitude of the speech samples and the sample to sample dependency.

Accounting for the performance improvement, a redundancy reducing technique in helping VPCM to utilize the

other redundancies (dependencies) in the speech waveform is desirable.

Before the discussion of a suitable redundancy reducing technique for VPCM, it is helpful to review the redundant information (dependencies) in the speech waveform.

3.3 Redundant information in the speech waveform

According to [17], the speech waveform dependencies can be classified into five categories:

(i) Non-uniform probability density function of the amplitude of the speech samples

(ii) Sample to sample dependency

(iii) Cycle to cycle dependency (periodicity)

(iv) Pitch period to pitch period dependency

(v) Inactivity factors (speech pauses)

Since only (iii) and (iv) are related to the development of the first extension of VPCM, only (iii) and (iv) are to be reviewed.

Human speech sounds are often classified into two distinct classes according to their mode of excitation. The first class is the "voiced" class of sounds which are produced by forcing air to flow from the lungs into the vocal cords so that the vocal cords vibrate in a relaxation oscillation. Such a vibration of the vocal cords, therefore, produces quasi-periodic pulses of air to excite the vocal tract. The interval between every two air pulses is referred to as the "pitch period". An example of a voiced speech waveform is shown in the voiced region V of Fig. 3.2. The

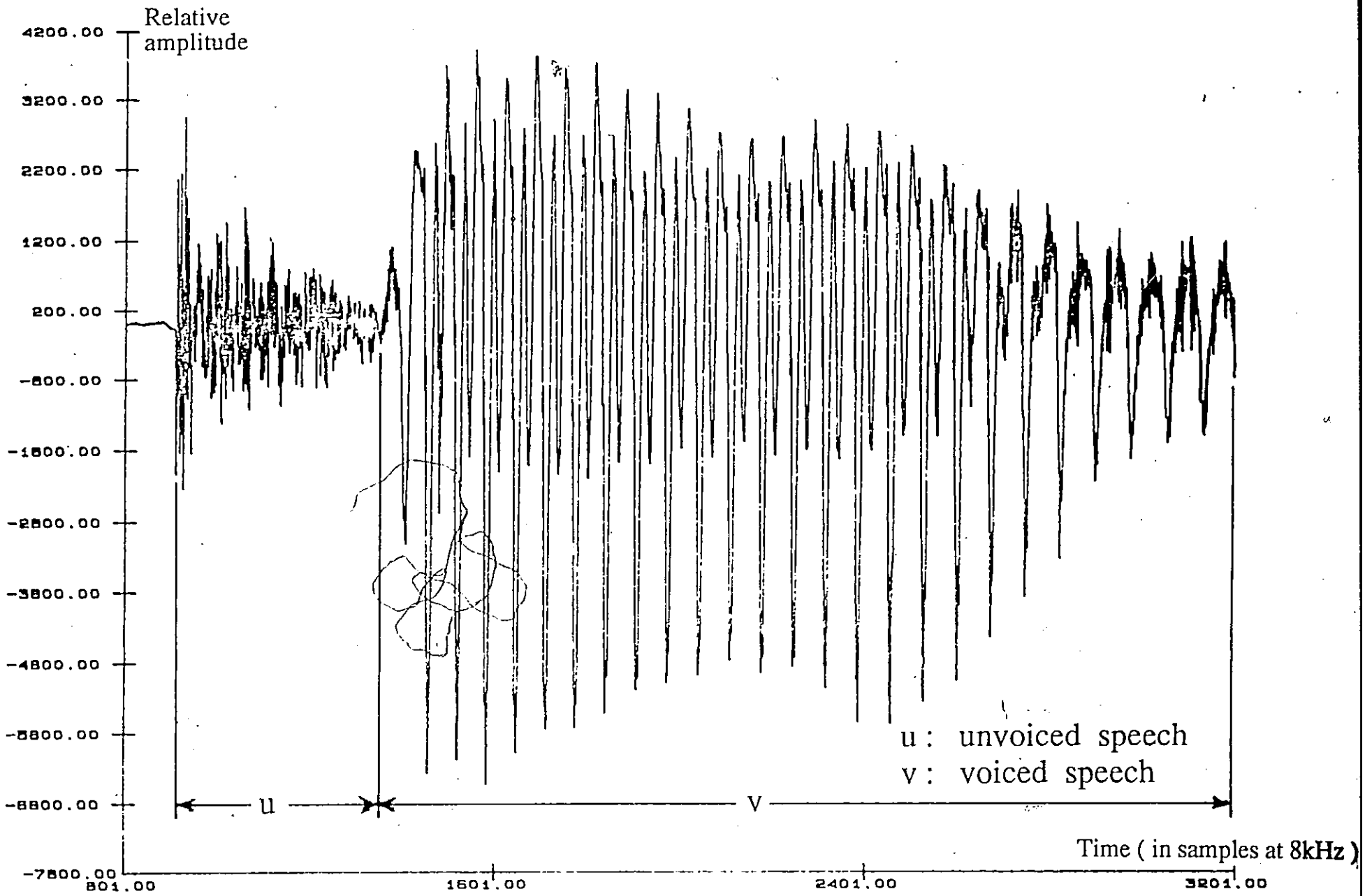


Fig. 3.2 A 300ms speech waveform from speaker [JT.24]

second class of sounds is the fricatives or "unvoiced" sounds. Unvoiced sounds occur as a result of continuous air flowing through the vocal tract constricted at some point to generate air turbulence. Because of the turbulent flow of air, unvoiced speech waveforms are much more random (noise-like) than voiced speech waveforms. An example of an unvoiced speech waveform is shown in the unvoiced region U of Fig. 3.2.

As the vocal tract movement is relatively slow compared to the pitch period (the rate of excitation of the air pulses), the shapes of the vocal tract are very similar to one another over several pitch periods. Thus, a voiced speech waveform displays a "long term" repetitive pattern as compared to the duration of a pitch period. This repetition in the speech waveform is referred to as the cycle to cycle dependency. In contrast, unvoiced sounds do not exhibit the cycle to cycle dependency because of the broad-spectrum noise-like excitation source (the turbulent flow of air).

Because of the quasi-periodic air pulses, the pitch periods are very close to each other. Such a similarity among the pitch periods of voiced sounds is referred to as pitch period to pitch period dependency.

3.4 Redundancy reducing technique for VPCM

As discussed in section 3.2.5, VPCM is not able to exploit the cycle to cycle dependency, the pitch period to pitch period dependency and the speech pauses. Therefore, a useful redundancy reducing technique for VPCM should be able to exploit at least one of these redundancies in the information.

As shown in Fig. 3.2 , the shape of a speech cycle (the speech waveform in a pitch period) is identical to that of the other speech cycles of the same voiced sound. Further, in the voiced region V, the duration of a speech cycle is almost the same as that of its adjacent cycles. However, the cycle to cycle energy variation is more than the variation of the duration and the shape of the speech cycles. Hence, the vectors (a speech cycle contains many vectors) which are one pitch period apart from each other are with similar "shape" but different energy. As the energy of a vector represents the square of its distance in the Euclidean space from the origin, the locations of the vectors in the Euclidean space are different from speech cycle to speech cycle. Furthermore, the LBG algorithm "recognizes" only the dependencies among vectors by "grouping" those vectors which are "close" to each other in the Euclidean space. The codewords being designed using the LBG algorithm (an iterative clustering algorithm) can hardly utilize the cycle to cycle dependency. Thus, VPCM considers and encodes each

speech cycle independently without exploiting the cycle to cycle dependency.

For the purpose of exploiting the cycle to cycle dependency, each speech cycle shall be normalized such that the vectors of a speech cycle are "located closely", in the Euclidean space, to those of the other speech cycles which belong to the same voiced sound. To achieve this, each speech cycle shall be normalized by its own maximum sample magnitude such that the highest energy vector from each of the speech cycles are guaranteed to be "placed closely" in the Euclidean space. Since unvoiced sounds do not exhibit the cycle to cycle and the pitch period to pitch period dependencies, they do not need to be normalized. Furthermore, since the normalization factors can be called the gain terms of the speech cycles, the normalization scheme together with VPCM can be called the Gain separated Voiced sound VPCM (GV-VPCM). A block diagram of GV-VPCM is shown in Fig. 3.3.

The normalized voiced sounds together with the unnormalized unvoiced sounds then act as the source input to the full search vector quantizer in GV-VPCM. The normalization factor (maximum sample magnitude) of each speech cycle together with the indexes of the nearest neighbours are transmitted to the receiver. At the receive end, the reconstructed normalized speech is expanded according to the received normalization factors.

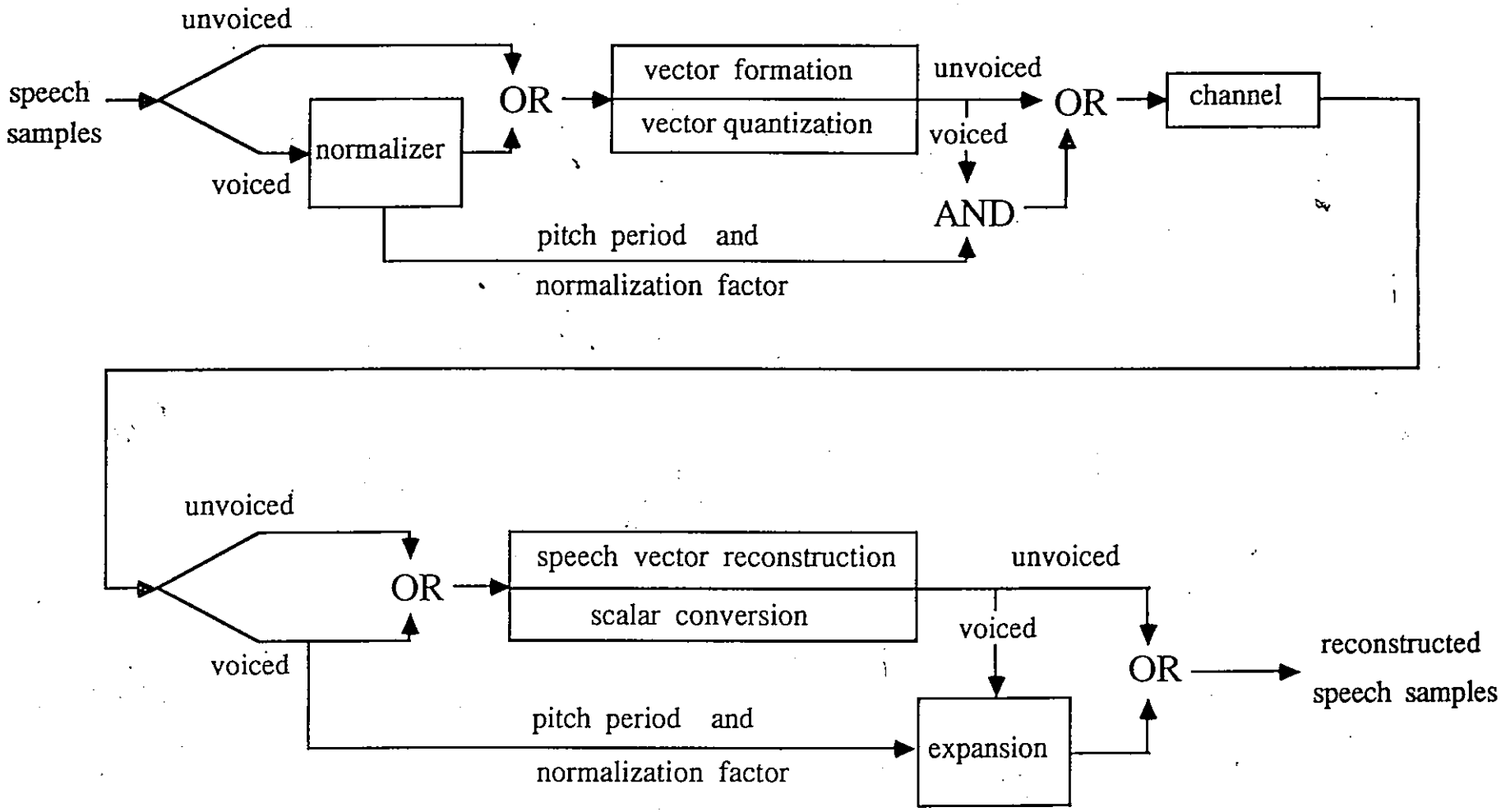


Fig. 3.3 Block diagram of GV-VPCM

Note that the sample amplitude of the normalized speech cycles from all voiced sounds is bounded by one. Hence, the vectors of the normalized voiced sounds are all located inside a bounded region in the Euclidean space. Therefore, the vector quantizer in VPCM is able to exploit not only the cycle to cycle dependency but also the possible dependencies that exist among the normalized speech cycles of different voiced sounds.

Finally, as GV-VPCM is the first attempt for the utilization of dependencies in VPCM, its development is discussed in detail in the following section. Section 3.5 is rather long. At first reading, I recommend that the reader concentrate only on the structure and on the simulation results of GV-VPCM and G-VPCM (a simplified version of GV-VPCM).

3.5 Development of GV-VPCM

Based on the previous discussion of GV-VPCM, there are three problems that arise from using GV-VPCM: voiced/unvoiced decision, pitch period detection, vector formation and the buffering scheme needed for fixed-rate transmission. Since unvoiced sounds are not normalized prior to vector quantization, the speech input to GV-VPCM must be classified as either voiced or unvoiced.

Pitch period detection must be employed for the selection of the speech cycles from voiced sounds. The maximum sample magnitude (normalization factor) of each speech cycle can be found only when the speech cycles are

defined. In addition, the normalization factor and the pitch period of each speech cycle must be transmitted to the receiver for speech reconstruction.

The problem of the vector formation is created because of the pitch period detection and the synchronization requirements. For the synchronization requirement, the encoded speech should be transmitted on a cycle by cycle basis. Each encoded speech frame (the encoded speech cycle) must provide the information about the normalization factor, the pitch period and the indexes of the nearest neighbours of the normalized speech cycle. Since the vector dimension is a fixed quantity for a given vector quantizer and the pitch period is time-varing, a non-integer number of vectors occurs inside a single pitch period. This is very undesirable as vector quantization cannot be applied to a "partial vector" (a vector with a dimension smaller than the defined vector dimension of a given vector quantizer). A method to deal with the resulting partial vector is to use the "overlapped speech cycle" shown in Fig. 3.4. Each encoded speech frame (the encoded overlapped speech cycle) must provide the information about <u>two</u> normalization factors, the overlapped pitch period and the indexes of the nearest neighbours of the overlapped speech cycle.

Finally, because the pitch periods typically last from 5ms to 20ms for men and from 2.5ms to 10ms for women [17], the overlapped speech cycles also result in variable-length. Therefore, certain buffering schemes would need to be

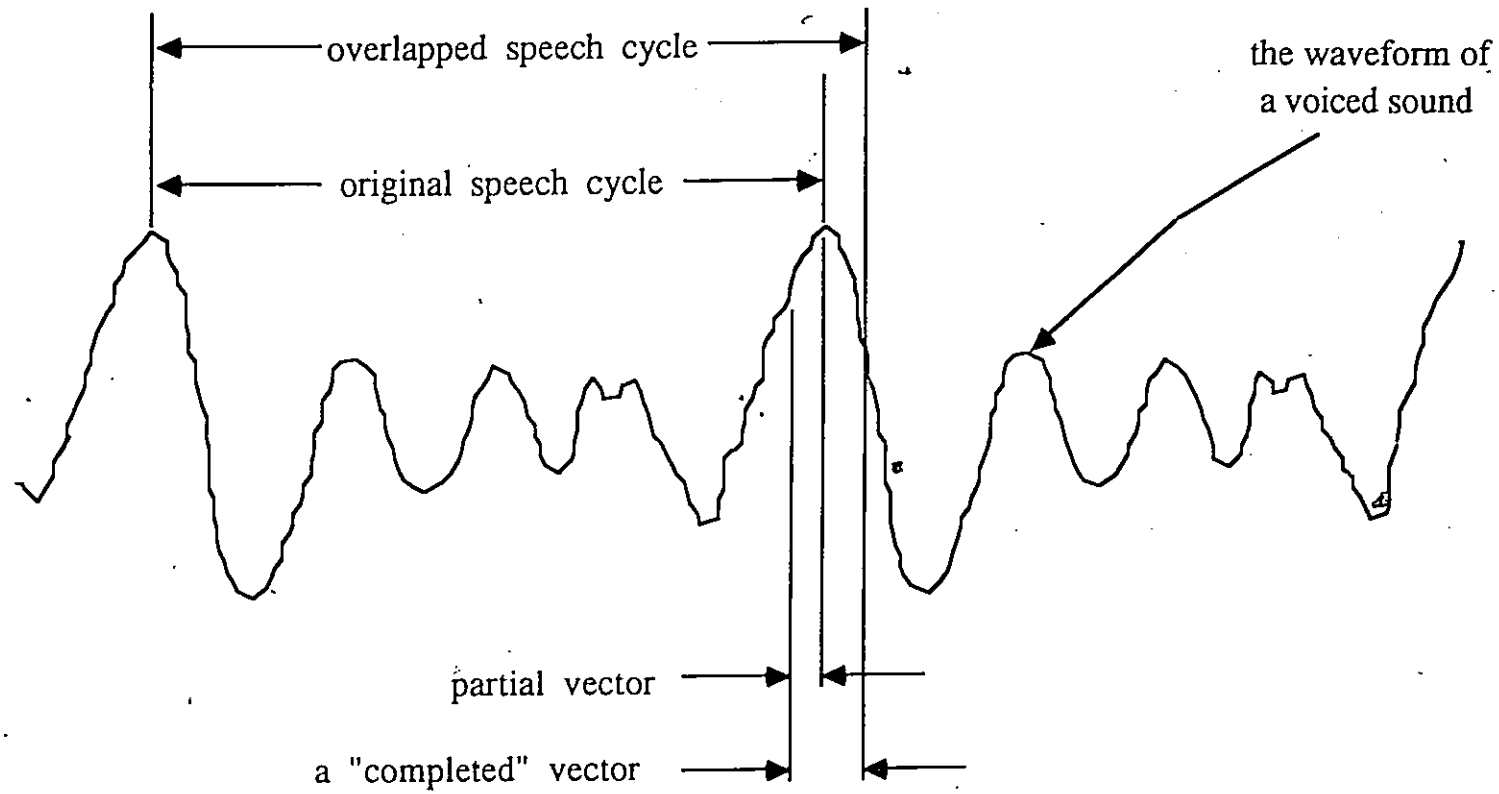


Fig. 3.4 The formation of the overlapped speech cycle

implemented if transmission is over a fixed-rate channel. In addition, since no side information (the normalization factors and the pitch periods) is required for unvoiced sounds, appropriate buffering schemes must be implemented even though the voiced sounds result in fixed-length overlapped speech cycles.

It is clear that GV-VPCM requires more computation than VPCM. The computations that are required for the voiced/unvoiced decision, the pitch period detection, the normalization factor detection, the cycle by cycle normalization, the buffering scheme for fixed-rate transmission and the cycle by cycle expansion are almost equivalent to that of the pitch-excited linear predictive coding [28]. Therefore, the total computation complexity (includes the computation of vector quantization) of GV-VPCM is greater than that of the pitch-excited linear predictive coding using vector quantization [9].

There are two questions arising before the full development of GV-VPCM. The first is the performance of GV-VPCM. Because of the huge computational requirement, GV-VPCM is practically useful only if it performs significantly better than VPCM. The second question is whether the computation complexity of GV-VPCM can be simplified without severe performance degradation. Only a simplified GV-VPCM is desired since the projected speech coder must be implementable.

3.5.1 Performance of GV-VPCM and VPCM in simulated speech

GV-VPCM is expected to outperform VPCM by exploiting the cycle to cycle dependency of the speech waveform. In this section, both GV-VPCM and VPCM were designed and tested on simulated speech in order to find quantitatively the performance improvement achieved. The performance of GV-VPCM and VPCM were measured by the SNR and the segmental signal to quantization noise ratio (SEGSNR). The SEGSNR is defined as

SEGSNR(dB) = the averaged value of the short-time SNR's(dB) over the non-silence portions of a test sequence.

where a short-time SNR(dB) is the SNR(dB) of a speech segment. The length of the speech segment is not standardized. In certain speech coding systems which encode speech on a frame by frame basis, the length of the speech segment is usually set to be equal to the length of the speech frame [22].

Unlike the SNR which tends to be unfair to low energy segments, the SEGSNR assigns more equitable weighting to high energy and low energy segments. Therefore, the SEGSNR is a better performance indicator than the SNR. Also, the SEGSNR is found to correlate better with the subjective quality of speech than the SNR [16].

As unvoiced sounds do not exhibit the cycle to cycle dependency, only the performance of GV-VPCM and VPCM on

voiced sounds are of interest. Thus, only voiced sounds are required as the training sequence for codebook design.

Usually, the speech synthesis model is able to generate only the voiced sounds while real speech always consists of voiced sounds, unvoiced sounds and silence. Simulated speech offers a direct generation of the required training sequence. Besides, the pitch period is one of the parameters for the synthesis of voiced sounds and it can be controlled such that no partial vector occurs inside any single pitch period. Thus, the problems of the pitch period detection, the vector formation and the buffering scheme, discussed at the beginning of section 3.5, are eliminated when using simulated speech.

The only requirement of the simulated speech is the repetitive pattern (the pitch period to pitch period and the cycle to cycle dependencies) and, the different energy assignments for different voiced sounds. A simplified voiced speech synthesizer is shown in Fig. 3.5 and the details of this model are given in Appendix A.

A simulated speech, generated by the model in Fig. 3.5, of about 1.35 seconds (10800 samples) was used for codebook design. The simulated speech, consisted of 15 different voiced sounds each with a fixed pitch period of 10ms (a typical pitch period of male speech), was equivalently sampled at 8kHz. Each voiced sound was generated according to its formant frequencies (F1, F2, F3) and amplitude Av. The duration of each voiced sound was 90ms (a typical voiced

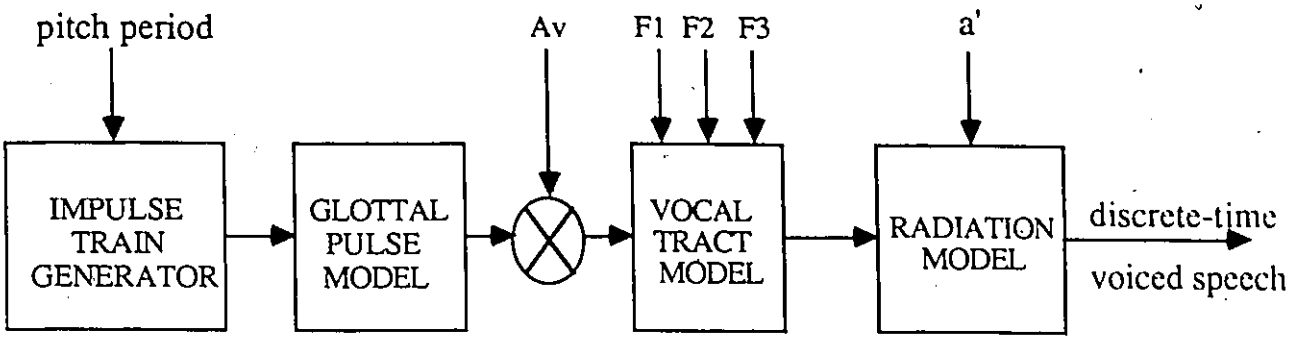


Fig. 3.5 Discrete-time model for voiced speech production

F1 : the first formant frequency
F2 : the second formant frequency
F3 : the third formant frequency
Av : amplitude control
a' : radius of the lips opening

PHONEME OF THE VOICED SOUNDS	F1 (Hz)	F2 (Hz)	F3 (Hz)	Av
I	350	2300	3340	90
E	440	2134	2780	100
EE	400	1825	2400	100
AE	750	1650	2255	125
A	755	1180	2610	110
ER	500	1500	2800	100
AW	550	900	2500	120
O	480	900	2490	150
U	325	795	2420	120
W	260	680	2320	40
R	515	1220	1950	70
IL	300	1300	3000	40
Y	320	2120	3100	90
LH	280	1450	1600	100
L	460	1500	2550	40

Table 3.2 The parameters of the voiced sounds for production. After [31]

sound lasts approximately 100ms [17]). For each of the voiced sounds, the duration of each speech cycle was set to be 10ms and the energy of the speech cycles was controlled by the radiation model with different a' (the radius of the "lips opening") varying from 0.5cm to 1.3cm. The parameter a' was varied from speech cycle to speech cycle.

The formant frequencies and the amplitude of the voiced sounds are shown in Table 3.2. These data are obtained from [31].

The vector dimension of GV-VPCM and VPCM was again selected as eight (k=8). However, the size of the codebook, N, was selected as 64 because of the relatively short training sequence (10800 samples). The 64 codewords of the 8-dimensional GV-VPCM and VPCM were designed using the LBG algorithm based on the unweighted squared error distortion measure together with the splitting initialization and a convergence threshold of 0.005. The normalization factors were not quantized. Finally, for the calculation of the SEGSNR, the length of the speech segments was set to be equal to the length of the speech cycles. Each of the speech segments was set to be 10ms. The simulation results are shown in Table 3.3.

k=8 N=64	SNR (dB)	SEGSNR(dB)
VPCM	11.65	9.65
GV-VPCM	13.52	14.27

Table 3.3 Simulation results (simulated speech)

From Table 3.3, the GV-VPCM outperforms the VPCM by an increase of 1.87 dB in SNR and an increase of 4.62 dB in SEGSNR. Since the SEGSNR correlates better with the subjective quality of speech, the "speech" output of the GV-VPCM is expected to "sound" better than that of the VPCM even though their SNR's differ only by 1.87 dB.

Considering the performance of the VPCM, the resulting SEGSNR being lower than the SNR, indicates that the resulting short-time SNR of the relatively low energy speech cycles is lower than that of the high energy speech cycles. On the contrary, the GV-VPCM achieving a higher SEGSNR than the SNR suggests that the resulting short-time SNR of the low energy speech cycles is comparable to that of the high energy speech cycles. Such an "equal treatment" between the low energy and high energy speech cycles offered by the GV-VPCM can be explained by recalling that the vectors of all the normalized speech cycles fall inside a bounded region in the Euclidean space (section 3.4). All of the codewords being designed using the LBG algorithm are located inside the same bounded region as the vectors of all the normalized speech cycles. Since all the speech cycles (voiced sounds) are of similar statistical characteristics (section 3.3), the low energy speech cycles are not "discriminated" as in the VPCM. Therefore, the dependencies among the normalized speech cycles of different voiced sounds (section 3.4) are shown to be utilized by the GV-VPCM.

Considering the utilization of the cycle to cycle dependency, the GV-VPCM does not appear to be efficient. Since the SNR is strongly influenced by the high energy speech segments, the relatively high energy speech cycles such as those from the sounds "AW" and "O" in Table 3.2 are assumed not to be significantly better coded by the GV-VPCM than by the VPCM. In other words, only a "little" improvement is achievable by using the normalization scheme to utilize the cycle to cycle dependency.

As a conclusion, GV-VPCM is better than VPCM in the sense that it is capable of utilizing the cycle to cycle dependency and the dependencies among different voiced speech cycles. Since the SNR does not indicate whether the low energy speech segments are well or poorly coded, the SEGSNR appears to be a more suitable performance measure for GV-VPCM. Finally, GV-VPCM is considered to be useful as it contributes to a 4.62 dB improvement in SEGSNR.

In the following sections, a study on simplifing GV-VPCM and the performance of the simplified GV-VPCM in real speech data are discussed.

3.5.2 The methods of simplifing GV-VPCM

As discussed in section 3.5, the main computation components that are required for the normalization scheme in GV-VPCM are respectively due to the voiced/unvoiced decision, the pitch period detection and the buffering scheme for the transmission of the encoded variable-length speech segments. The simplification of GV-VPCM can be

realized either by reducing the computation complexity of the stated requirements or by simplifing the normalization scheme itself.

From the preliminary results of GV-VPCM (section 3.5.1), the SNR improvement achieved by the utilization of the cycle to cycle dependency using the normalization scheme is not significant. Therefore, the required normalization scheme can be modified such that only the dependencies among the normalized speech cycles of different voiced sounds are utilized by GV-VPCM. As a conclusion, the simplification of G-VPCM is focussed on the normalization scheme.

Recall from section 3.3, the waveform of a voiced sound is approximately a "periodic" signal which resulted from one single speech cycle repeating itself periodically. This is true only for a voiced sound that exhibits strong pitch period to pitch period dependency. However, because of the relaxation oscillation of the vocal cords, strong pitch period to pitch period dependency can always be expected from voiced sounds. Only the voiced sounds that are spoken by nervous or frightened speakers whose vocal cords change the oscillation frequency rapidly, exhibit weak pitch period to pitch period dependency. Therefore, the normalization scheme, which is required to normalize the voiced sounds cycle by cycle, can be implemented using fixed-length speech segments with a segment-length approximately equal to the duration of a speech cycle. Since the pitch periods change from speaker to speaker, the fixed-length speech segments

may cover more than one speech cycle for "high pitch" speakers and less than one speech cycle for "low pitch" speakers. Therefore, the cycle to cycle dependency may not be utilized as much as the original normalization scheme. However, the fixed-length speech segments normalization scheme is remarkable as it does not require the pitch period detection. The selection of the length of the fixed-length segment will be given in section 3.5.3.

As discussed in section 3.5, the fixed-length speech segment normalization scheme does require the voiced/unvoiced decision and appropriate buffering scheme if it is applied only to voiced sounds. The voiced/unvoiced decision and the buffering scheme can be eliminated only if unvoiced sounds are also normalized in the same manner as the voiced sounds. However, the normalization of unvoiced sounds may lead to performance degradation of GV-VPCM. Therefore, similar to the fixed-length speech segment normalization scheme, the normalization of unvoiced sounds is another compromise between the computation complexity and the performance of GV-VPCM.

Unvoiced sounds are noise-like signals, and remain noise-like after normalization. By forcing their vectors to be placed close to the vectors of voiced sounds in the Euclidean space, the LBG algorithm may not be able to recognize and exploit the cycle to cycle dependency (section 3.4). Therefore, a GV-VPCM that normalizes voiced sounds as

well as unvoiced sounds may not be able to utilize the cycle to cycle dependency.

Not surprisingly, there are advantages from normalizing the unvoiced sounds. Basically, there are two advantages: computational reduction and dependency utilization. The computational reduction is the saving of the voiced/unvoiced decision and buffering scheme. The dependency utilization is: the vectors of the normalized voiced and unvoiced sounds are all bounded by the same region in the Euclidean space, thereby allows the vector quantizer to utilize any dependency that exists between the vectors of the normalized voiced and unvoiced sounds (section 3.4). For instance, the waveform envelopes of those voiced sounds with low F1's and high F2's, F3's (such as the "I" in Table 3.2 and the waveform in the right-hand side of the region V in Fig. 3.2) are noise-like signals [30]. After they are normalized, their envelopes are still noise-like signals. Therefore, their normalized vectors are similar to the normalized unvoiced speech vectors as they all come from a source with a noise-like waveform envelope. Hence, there are certain dependencies that exist between the normalized unvoiced and the decribed voiced sounds.

As a summary, because of the pitch period to pitch period dependency, the pitch period detection can be eliminated by using fixed-length speech segment normalization scheme. The price to pay for that is the resulting weaker capability in utilizing the cycle to cycle

dependency. For the elimination of the voiced/unvoiced decision and buffering scheme, the normalization of unvoiced sounds appears to have two conflicting effects on the resulting performance. The first is the performance degradation which results because the cycle to cycle dependency are not effectively utilized. The second effect which is caused by the utilization of the dependencies that exist between the normalized voiced and unvoiced speech vectors, tends to improve the resulting performance. If the pitch period detection, the voiced/unvoiced decision and buffering scheme are all eliminated through the methods stated above, the simplified GV-VPCM will not be able to utilize effectively the cycle to cycle dependency. However, the simplified GV-VPCM is able to utilize the dependencies between the normalized voiced and unvoiced speech vectors. These dependencies are not possibly utilized by VPCM and GV-VPCM.

As a conclusion, the selected method of simplifing GV-VPCM is a modified normalization scheme which normalizes the speech input (voiced and unvoiced) over fixed-length speech segments. This method allows a significant reduction in computation complexity and counteracts the resulting performance degradation by exploiting the dependencies which are not possibly utilized using the original normalization scheme. Moreover, as unvoiced sounds are also normalized, the simplified GV-VPCM shall be called the Gain separated VPCM (G-VPCM). Similar to GV-VPCM, the structure of G-VPCM

is the same as GV-VPCM except the length of the segments is a constant and unvoiced sounds are also normalized. The source input to the full search vector quantizer in G-VPCM is the normalized speech which includes voiced and unvoiced sounds. A block diagram of G-VPCM is shown in Fig. 3.6.

3.5.3 Selection of the segment-length of G-VPCM

As the pitch period of speakers ranges from 2.5ms to 20ms, an arbitrary fixed-length segment covers either more than one speech cycle or less than one speech cycle. The length of the required segments, therefore, can be set arbitrarily between 2.5ms and 20ms.

Since the normalization factors (the side information of each segment) must be transmitted to the receiver, the transmission rate of the normalization factors will be high if the length of the segments is very short. For the purpose of minimizing this transmission rate, the length of the segments should be as long as possible.

However, the longer the segments, the longer the delay time of the system (encoder and decoder) and the higher the probability of the occurrence of the transitions of voiced/unvoiced sounds and of different voiced sounds. Also, the energy of voiced sounds varies as much as 300% of one another (Table 3.2) and different energy voiced sounds

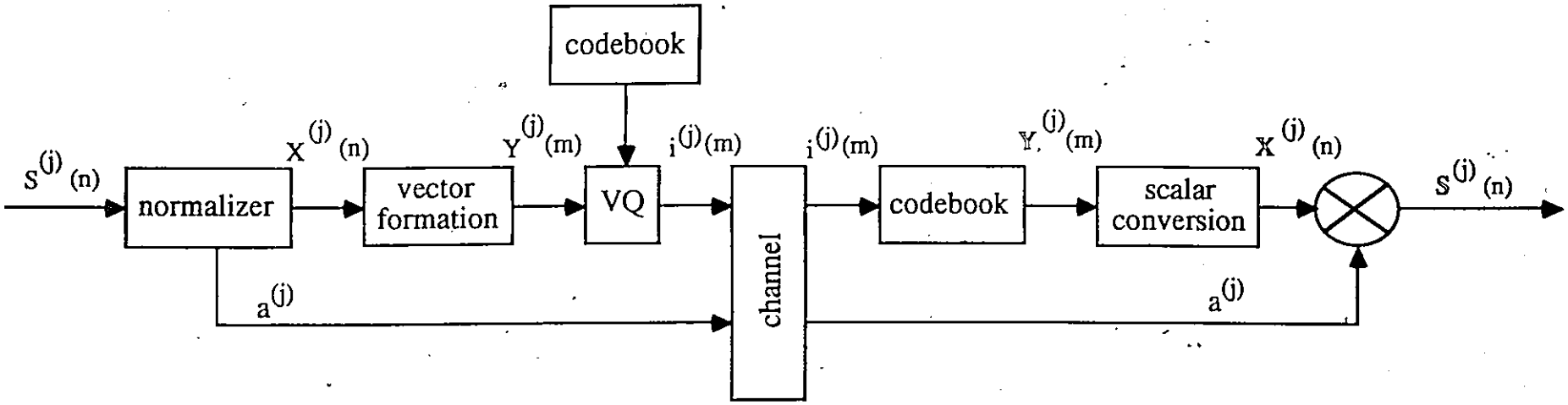


Fig. 3.6 Block diagram of G-VPCM

VQ : the encoding process of the full search vector quantizer

$S^{(j)}(n)$ = the n^{th} sample of the j^{th} speech segment ; n=1,...,L

L = the length of the speech segments

$a^{(j)}$ = the maximum sample magnitude of the j^{th} speech segment

$X^{(j)}(n)$ = the n^{th} sample of the j^{th} normalized speech segment

$$= \frac{S^{(j)}(n)}{a^{(j)}}$$

$Y^{(j)}(m)$ = the m^{th} vector of the j^{th} normalized speech segment ; m=1,...,M

$$M = \frac{L}{k}$$ where k is the vector dimension

$i^{(j)}(m)$ = the index of the nearest neighbor of $Y^{(j)}(m)$

$\mathbb{Y}^{(j)}(m)$ = the reconstructed $Y^{(j)}(m)$

$\mathbb{X}^{(j)}(n)$ = the reconstructed $X^{(j)}(n)$

$\mathbb{S}^{(j)}(n)$ = the reconstructed $S^{(j)}(n)$

require different normalization factors. A single normalization factor is not adequate for the normalization of the segments that include the stated transitions. Hence, to account for minimizing the occurrence of the stated transitions in segments and the coder delay, the length of the segments should be as short as possible.

Based on the above discussion, a length of 10ms is selected to be suitable for the segments of G-VPCM. It is short enough for the classification of voiced/unvoiced/silence speech [23], thereby the 10ms segments almost do not include the transitions of voiced/unvoiced sounds and different voiced sounds. In addition, the transmission rate of the side information is only one parameter for every 10ms.

3.5.4 Quantization of the normalization factors

For the purpose of computation, the encoding process of the vector quantizer in G-VPCM must be handled by digital circuits. Likewise, the selection of the normalization factors and the normalization of the corresponding speech segments must be done digitally. Therefore, the speech samples must be digitized prior to normalization.

As is generally accepted in the literature and in practice, linear quantization (A/D) is the only digitization scheme available for speech samples. Therefore, the normalization factors must, firstly, be quantized through a given A/D. By learning the statistical charateristics of the normalization factors, certain non-linear quantization

schemes that perform better (less quantization noise) than the linear quantizer (A/D) are possible.

However, the transmission rate is only 1 normalization factor for every 10ms and the A/D of the normalization factors does not require any extra computation which is necessary when using the non-linear quantization schemes, minimizing the required transmission rate through a non-linear quantizer is not a good solution. After all, a saving of 1 bit in quantizing the normalization factors is equivalent to a saving in only 100 bits/s of the corresponding transmission bit rate. Such a saving in the transmission bit rate can be neglected in medium-band (transmission bit rate ranges from 8 kb/s to 16 kb/s) speech coding systems. Thus, only the linear quantizer (A/D) is good enough for the quantization of the normalization factors.

3.5.5 Transmission bit rate of G-VPCM

From the previous section, the speech samples are digitized for digital computation. For the purpose of computational accuracy, a minimum wordlength of either 12 bits or 16 bits is necessary. Therefore, the speech samples should be digitized either through a 12 bits A/D or a 16 bits A/D.

As the wordlength of most of the Digital Signal Processors (DSP) and micro-processors is 16 bits, a 16 bits A/D is assumed for the quantization of the normalization

factors. Therefore, the transmission bit rate of the normalization factors is 16 bits for every 10ms.

As discussed in section 3.2.1, the transmission bit rate of the vector quantizer in G-VPCM, R, is

$$R = \frac{\log_2 N}{k} \cdot f_s \quad \text{bits/sec}$$

Therefore, the overall transmission bit rate of G-VPCM, R_G, is

$$R_G = \frac{\log_2 N}{k} \cdot f_s + 1600 \quad \text{bits/sec} \qquad (28)$$

3.5.6 Performance of G-VPCM and VPCM in real speech

As discussed in section 3.2, an interesting transmission rate of VPCM is 1 bit/sample; the transmission rate of the vector quantizer in G-VPCM, therefore, is also 1 bit/sample. Likewise, the distortion measure, the vector dimension and the size of the codebook in G-VPCM are the same as that in VPCM. As a result, the parameters of the desired G-VPCM and VPCM are shown in Table 3.4.

Similar to the simulation of section 3.5.1, the performance of the G-VPCM and the VPCM were measured by the SNR and the SEGSNR. The segment-length for the calculation of the SEGSNR was set equal to the length of the speech segments of the G-VPCM. The classification of the silence segments (speech segments include silence speech only) was carried out through the log energy measure

	G-VPCM	VPCM
Distortion measure	Unweighted squared error distortion measure	
Sampling frequency	8 kHz	
Length of speech segments	10ms	N/A
Number of codewords	256	
Transmission rate of the vector quantizer	1 bit/sample	
Transmission bit rate of the normalization factors	1.6 kb/s	N/A
Overall transmission bit rate	9.6 kb/s	8 kb/s

Table 3.4 Parameters of the selected G-VPCM and VPCM

k=8 N=256	Train. mode		Test. mode	
	SNR (dB)	SEGSNR (dB)	SNR (dB)	SEGSNR (dB)
VPCM	11.60	10.41	12.24	9.96
G-VPCM	10.96	14.60	12.68	14.55

Table 3.5 Simulation results (real speech)

(Appendix B). The training sequence and the test sequence described in section 3.2.4 were used for the design and the test of the codebooks of the G-VPCM and the VPCM. The codebooks were designed using the LBG algorithm based on the unweighted squared error distortion measure together with the splitting initialization and a convergence threshold of 0.005. The simulation results are shown in Table 3.5.

The results obtained with respect to the training sequence and the test sequence are shown in the columns Train.mode and Test.mode, respectively. These results show the usefulness of the designed codebooks. Since the statistical characteristics of the normalized speech (the source input to the vector quantizer in G-VPCM) are different from those of the original speech (the source input to the vector quantizer in VPCM), the training sequence, which is long enough for the codebook design of the VPCM (section 3.2.4), may not be long enough for the G-VPCM.

From Table 3.5, the resulting SNR of the G-VPCM that is obtained using the training sequence is very close to the one obtained using the test sequence. Therefore, according to section 2.3, the training sequence is also long enough for the codebook design of the G-VPCM.

From the resulting SNR, the G-VPCM is inferior to the VPCM by 0.64 dB with respect to the training sequence. Using the test sequence, the resulting SNR of the G-VPCM is superior to the VPCM by 0.44 dB. Therefore, based on the

resulting SNR, the G-VPCM is not better than the VPCM. Hence, the relatively high energy speech segments (typically voiced speech segments) are coded, approximately, equally "well" by the G-VPCM and the VPCM (section 3.5.1). Hence, the capability of the G-VPCM is about the same as that of the VPCM in utilizing the cycle to cycle dependency (section 3.5.1).

Considering the resulting SEGSNR, the G-VPCM is superior to the VPCM by 4.19 dB in Train.mode and by 4.59 dB in Test.mode. Therefore, the relatively low energy speech segments (typically unvoiced sounds) are better reconstructed by the G-VPCM. This is expected because the dependencies between the normalized voiced and unvoiced sounds are exploited by G-VPCM but not by VPCM (section 3.5.2).

As a conclusion, the G-VPCM achieves a SEGSNR higher than that of the VPCM by utilizing the dependencies between the normalized voiced and unvoiced sounds while disregarding the cycle to cycle dependency. Therefore, the speech output of the G-VPCM is assumed to be subjectively better than that of the VPCM.

3.6 Conclusion and discussion of GV-VPCM and G-VPCM

Based on the simulation results from section 3.5.1, GV-VPCM does outperform VPCM, in the sense of the higher SNR and SEGSNR resulting, by exploiting the cycle to cycle dependency and the dependencies among the normalized speech cycles of different voiced sounds. However, the extra computational requirement that results from using the normalization scheme in GV-VPCM is very demanding.

G-VPCM, a simplified version of GV-VPCM, is then introduced to attempt to reduce computational complexity. As desired, the computation of the modified normalization scheme in G-VPCM is substantially less than that of the original normalization in GV-VPCM. The price to pay for this computational reduction is the lack of the capability in utilizing the cycle to cycle dependency. However, based on the simulation results from section 3.5.6, G-VPCM is still able to have a SEGSNR higher than that of VPCM by exploiting the dependencies between the normalized voiced and unvoiced sounds. Therefore, without the direct comparison between GV-VPCM and G-VPCM (both of the same vector dimension and codebook size), one can conclude that GV-VPCM is superior to G-VPCM with respect to the SNR measure. As the normalization scheme in G-VPCM is simpler, in the sense of computation complexity, than that in GV-VPCM and G-VPCM is shown to be better than VPCM by an increase of approximately 5 dB in SEGSNR, G-VPCM is chosen to replace GV-VPCM.

In the subsequent sections, an adaptive technique is introduced to attempt to "recover" the capability of the cycle to cycle dependency utilization of G-VPCM.

3.7 Adaptive technique for G-VPCM

Recall that through the normalization of the segments of voiced and unvoiced sounds, G-VPCM losses the capability of utilizing the cycle to cycle dependency but gains in exploiting the dependencies between the normalized voiced and unvoiced sounds. As only the voiced sounds whose waveform envelopes are noise-like exhibit dependencies with unvoiced sounds after normalization, G-VPCM is able to exploit these dependencies by using another codebook for only the normalized segments of these voiced sounds and unvoiced sounds. If the input speech of G-VPCM can be, somehow, classified and separated in a way that only unvoiced sounds and the voiced sounds with noise-like waveform envelopes belong to one class while the rest of the voiced sounds belong to another class(es), G-VPCM will be able to utilize the dependencies between the normalized voiced sounds (those with noise-like waveform envelopes) and unvoiced sounds. Moreover, the cycle to cycle dependency can be utilized by using different codebooks for different classes of speech.

The adaptive technique developed by Cuperman [3,22] appears to be suitable for the realization of the stated classification and separation schemes. As the adaptation of this technique is realized by making use of a set of fixed

codebooks rather than a time-varying codebook, this technique is more meaningful to be called the fixed adaptive technique. This technique is described in the following section.

3.7.1 Fixed adaptive technique

The fixed adaptive technique is performed on a segment by segment basis. Each speech segment is classified, based on the energy and the unit delay normalized autocorrelation coefficent of the current speech segment, into one of three classes. Associated with these three classes are three distinct codebooks, all of which are optimized for the particular class to which they belong. The vectors in the current speech segment are coded with the codebook which corresponds to this speech segment. The class identification, which requires only 2 bits, is transmitted to the receiver as side information of the current speech segment.

The classifier (the classification of the speech segments) in [22] is the following:

Any speech segment whose $r(1)$ and σ^2 are

$r(1) \geq 0.94$	and $\sigma^2 \geq 0.25\sigma_0^2$	belongs to Class 1
$0.94 > r(1) \geq 0.7$	and $\sigma^2 \geq 0.25\sigma_0^2$	belongs to Class 2
$r(1) < 0.7$	or $\sigma^2 < 0.25\sigma_0^2$	belongs to Class 3

where $r(1)$ = the unit delay normalized autocorrelation coefficent of the current speech segment.

σ^2 = the energy of the current speech segment.

σ_0^2 = the energy of all the training sequences for the three codebooks.

The design of this classifier was based on a sequence of 1029 segments of speech, sampled at 8 kHz and digitized by a 12 bit A/D converter. The length of each segments was 7.5ms (60 samples). The boundary values, 0.7 and 0.94 of r(1) and $0.25\sigma_0^2$ of σ^2, were determined such that each class contained approximately the same number of speech segments. Hence, the codebooks of the three classes have equal chance for the encoding of any input vector.

As a result, Class 1 ($r(1) \geq 0.94$ and $\sigma^2 \geq 0.25\sigma_0^2$) typically consists of voiced sounds and certain highly correlated and high energy unvoiced sounds such as plosives; Class 2 ($0.94 > r(1) \geq 0.7$ and $\sigma^2 \geq 0.25\sigma_0^2$) mostly consists of voiced sounds; Class 3 ($r(1) < 0.7$ or $\sigma^2 < 0.25\sigma_0^2$) typically consists of unvoiced sounds and, certain low correlated voiced sounds or low energy silence speech [22].

Hence, the cycle to cycle dependency and the dependencies among the speech cycles of different voiced sounds can be utilized by the G-VPCM together with this fixed adaptive technique, for the input speech which belongs to Class 1 and Class 2. Furthermore, because of the noise-like characteristic of unvoiced sounds and the voiced sounds with noise-like waveform envelopes, the r(1), the correlation of adjacent samples, of these voiced sounds can be expected to be close to that of unvoiced sounds (examples are given at the end of Appendix B). Hence, the voiced sounds with noise-like waveform envelopes can be expected to be classified into Class 3. Therefore, the dependencies

between the normalized voiced sounds with noise-like waveform and unvoiced sounds can be utilized by the G-VPCM together with this fixed adaptive technique, for the input speech which belongs to Class 3.

Although more computation is required by the r(1) and σ^2 than by the normalization scheme in G-VPCM, this is negligible when compared to the computations required for the full search vector quantizer. About the transmission bit rate of the class identification, only 200 bits/s (2 bits for every 10ms) is required when applied in G-VPCM. This is almost negligible when compared to the 9.6 kb/s G-VPCM.

To make this fixed adaptive technique applicable to G-VPCM with a segment-length of 10ms, the given boundary values, which are chosen based on the 1029 speech segments with each segment of 7.5ms, may need adjustment for other speech signals. Therefore, the application of the fixed adaptive technique in G-VPCM requires the given boundary values of r(1) and σ^2 to be verified by the training sequence. If the resulting classes of speech consist of different types of speech (voiced, unvoiced or silence) from what are desribed previously, a new set of the boundary values of r(1) and σ^2 is necessary.

3.7.2 Fixed adaptive technique for G-VPCM

Since the correlation of adjacent samples is a very good measure to distinguish between the voiced sounds with smooth waveform envelopes and unvoiced sounds including the voiced sounds with noise-like waveform envelopes, a

classifier employing only the r(1) is adequate for G-VPCM. Hence, the desired adaptive technique, except for the classifier, is the same as the described fixed adaptive technique.

As a result, the desired classifier is the following: Any speech segment whose r(1) is

$r(1) \geq 0.94$	belongs to	Class 1
$0.94 > r(1) \geq 0.7$	belongs to	Class 2
$r(1) < 0.7$	belongs to	Class 3

The boundary values of r(1) were obtained (Appendix B), based on the training sequence desribed in section 3.2.4, with the segment-length of 10ms (80 samples).

Based upon these values of r(1), Class 1 ($r(1) \geq 0.94$) typically consists of voiced sounds. Silence speech (Appendix B) also belongs to Class 1. Class 2 ($0.94 > r(1) \geq 0.7$) typically consists of voiced sounds and, certain unvoiced sounds and silence speech (Appendix B). Class 3 typically consists of unvoiced sounds. Certain low correlated voiced sounds such as those with noise-like waveform envelopes also belong to Class 3 (section 3.7.1 and Appendix B). Therefore, the dependencies in each class of speech are expected to be utilized by the G-VPCM together with this fixed adaptive technique. Finally, for the sake of identification, the G-VPCM that employs this adaptive technique will be called the fixed Adaptive G-VPCM (AG-VPCM).

It should be emphasized that this fixed adaptive technique can also be applied in VPCM. As will be discussed in the next section, both G-VPCM and VPCM are made adaptive using this fixed adaptive technique and, the adaptive VPCM is called A-VPCM. The block diagrams of A-VPCM and AG-VPCM are shown in Fig. 3.7 and Fig. 3.8, respectively.

3.8 Performance of A-VPCM and AG-VPCM in real speech

A-VPCM is included in order to find out how the dependencies in each class of speech are utilized by AG-VPCM. The parameters of the two adaptive systems, except for their transmission bit rates and their total number of codewords (256 codewords for each class of speech), are the same as those of the corresponding non-adaptive systems. In summary, the parameters of A-VPCM and AG-VPCM are shown in Table 3.6.

The training sequence described in section 3.2.4 was used for the design of the codebooks for all the three classes. The training sequence was divided into three individual sequences by the classifier described in section 3.7.2. Each of these individual sequences, then, became the training sequence for the codebook design of the corresponding class. However, the training sequence, consisted of only 33840 samples, for Class 3 was found to be too short for the design of the 8-dimensional codebook of 256 codewords. Some other sequences, spoken by the same speakers of the original training sequence also belong to

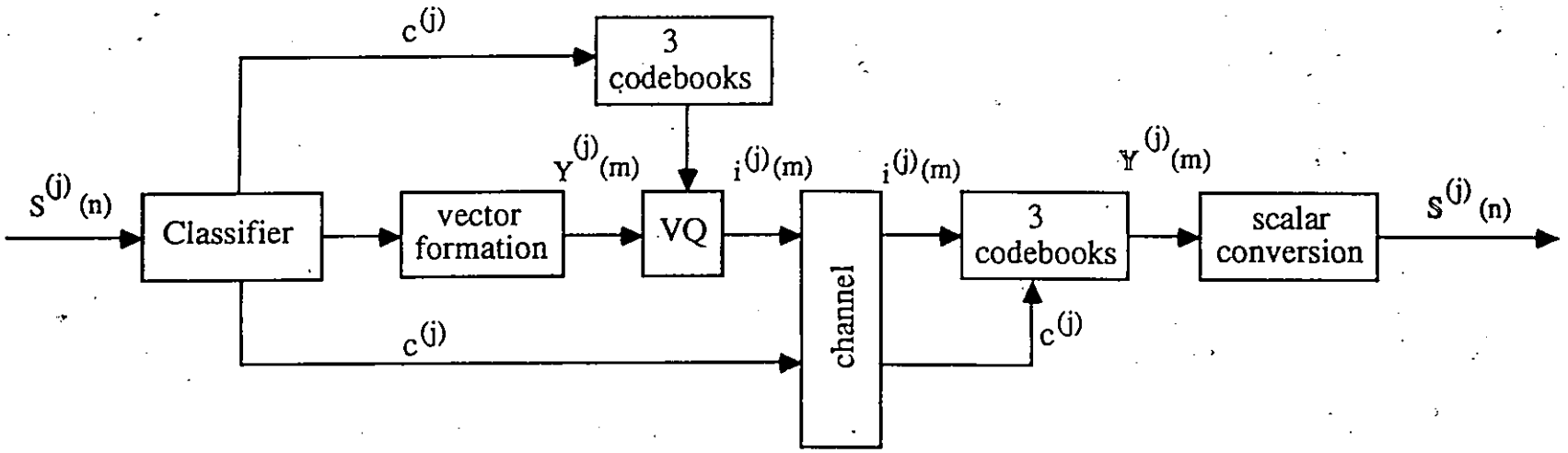


Fig. 3.7 Block diagram of A-VPCM

VQ : the encoding process of the full search vector quantizer

$S^{(j)}(n)$ = the n^{th} sample of the j^{th} speech segment ; n=1,...,L

L = the length of the speech segments

$c^{(j)}$ = the class identification of $S^{(j)}(n)$

$Y^{(j)}(m)$ = the m^{th} vector of the j^{th} speech segment ; m=1,...,M

$M = \frac{L}{k}$ where k is the vector dimension

$i^{(j)}(m)$ = the index of the nearest neighbor of $Y^{(j)}(m)$

$\tilde{Y}^{(j)}(m)$ = the reconstructed $Y^{(j)}(m)$

$\tilde{S}^{(j)}(n)$ = the reconstructed $S^{(j)}(n)$

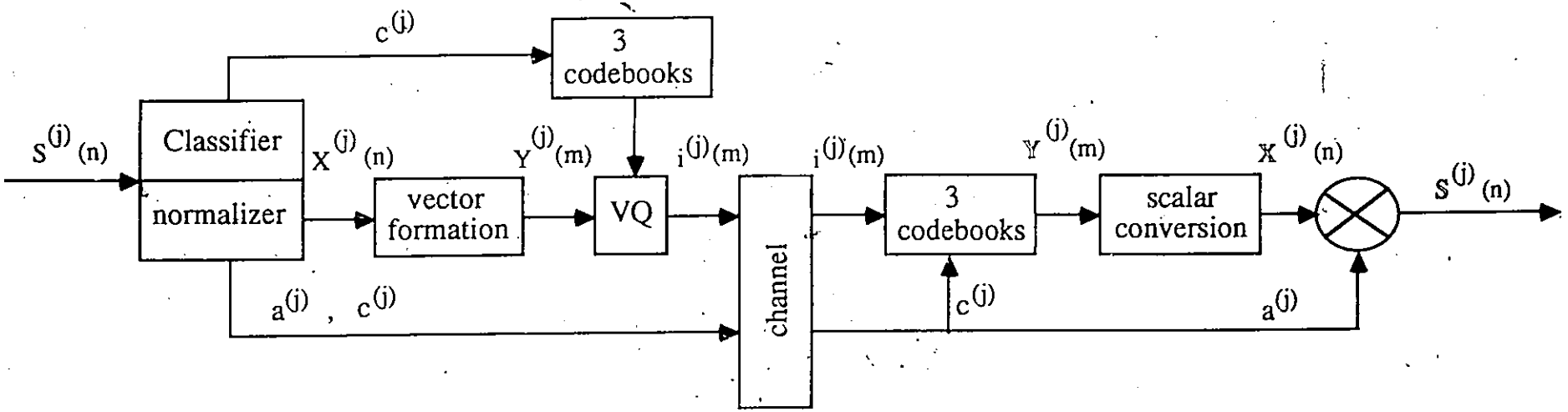


Fig. 3.8 Block diagram of AG-VPCM

VQ : the encoding process of the full search vecto quantizer

$s^{(j)}(n)$ = the n^{th} sample of the j^{th} speech segment ; n=1,...,L

L = the length of the speech segments

$c^{(j)}$ = the class identification of $s^{(j)}(n)$

$a^{(j)}$ = the maximum sample magnitude of the j^{th} speech segment

$x^{(j)}(n)$ = the n^{th} sample of the j^{th} normalized speech segment

$$= \frac{s^{(j)}(n)}{a^{(j)}}$$

$Y^{(j)}(m)$ = the m^{th} vector of the j^{th} normalized speech segment ; m=1,...,M

$$M = \frac{L}{k}$$ where k is the vector dimension

$i^{(j)}(m)$ = the index of the nearest neighbor of $Y^{(j)}(m)$

$\hat{Y}^{(j)}(m)$ = the reconstructed $Y^{(j)}(m)$

$\hat{x}^{(j)}(n)$ = the reconstructed $x^{(j)}(n)$

$\hat{s}^{(j)}(n)$ = the reconstructed $s^{(j)}(n)$

	AG-VPCM	A-VPCM
Distortion measure	Unweighted squared error distortion measure	
Sampling frequency	8 kHz	
Length of speech segments	10ms	
Number of codebooks	3	
Number of codewords in each codebook	256	
Transmission rate of the vector quantizer	1 bit/sample	
Transmission bit rate of the normalization factors	1.6 kb/s	N/A
Transmission bit rate of the class identification	0.2 kb/s	
Overall transmission bit rate	9.8 kb/s	8.2 kb/s

Table 3.6 Parameters of the selected AG-VPCM and A-VPCM

Class 3, were combined into the sequence of 33840 samples. The resulting sequence of 80240 samples was used for the codebook design for Class 3. There were no problems about the length of the training sequences for Class 2 and Class 1. The length of the training sequences for Class 2 and Class 1 were of 103520 samples and of 169280 samples, respectively.

Similarly, the test sequence described in section 3.2.4 was divided, in the same manner as the training sequence, into three individual test sequences. The test sequences for Class 1, Class 2 and Class 3 were of 48800 samples, 18240 samples and 9200 samples, respectively.

Each of the three codebooks were designed using the LBG algorithm based on the unweighted squared error distortion measure together with the splitting initialization, and a convergence threshold of 0.005. The simulation results are shown in Table 3.7.

As the usefulness of each designed codebook can be measured only from the simulation results of each class, only the simulation results of the overall adaptive systems in Test.mode are necessary for the evaluation of the performance of A-VPCM and AG-VPCM. The discussion of the simulation results is given in the following section.

3.8.1 Discussion of the simulation results

The simulation results of each class will be discussed separately. Subsequently, a comparison between the performance of A-VPCM and AG-VPCM is given.

		Train. mode		Test. mode	
		SNR (dB)	SEGSNR (dB)	SNR (dB)	SEGSNR (dB)
Class 1 k=8 N=256	A-VPCM	17.11	15.65	15.19	14.32
	AG-VPCM	16.73	20.07	16.20	18.08
Class 2 k=8 N=256	A-VPCM	11.73	10.24	11.67	8.28
	AG-VPCM	11.75	11.86	12.09	12.31
Class 3 k=8 N=256	A-VPCM	9.93	3.31	2.46	2.13
	AG-VPCM	7.42	6.24	5.22	5.28
Whole system k=8 N=256x3	A-VPCM	N/A		12.56	11.40
	AG-VPCM			13.32	15.15

Table 3.7 Simulation results of A-VPCM and AG-VPCM

(i) Class 1

Since the difference between the resulting SNR of AG-VPCM in Train.mode and in Test.mode is only 0.53 dB, the training sequence of 169280 samples is concluded to be long enough for the codebook design of AG-VPCM. However, the resulting SNR of A-VPCM in Train.mode and in Test.mode differs by 1.92 dB. A training sequence longer than 169280 samples should be used for the A-VPCM.

Considering the resulting SNR, AG-VPCM is superior to A-VPCM by 1.01 dB in Test.mode and is inferior to A-VPCM by 0.38 dB in Train.mode. Therefore, the cycle to cycle dependency is not effectively utilized by AG-VPCM. The reason for that is the existence of the unwanted signals (unvoiced sounds and silence speech) which do not exhibit any cycle to cycle dependency.

Considering the resulting SEGSNR, AG-VPCM is superior to A-VPCM by 3.76 dB in Test.mode and by 4.42 dB in Train.mode. Therefore, the dependencies among the normalized speech cycles of different voiced sounds are successfully utilized by AG-VPCM.

(ii) Class 2

The difference between the resulting SNR of AG-VPCM or A-VPCM in Train.mode and in Test.mode is less than 0.5 dB. Therefore, the 103520 samples training sequence appears to be long enough for the codebook design of both AG-VPCM and A-VPCM.

Considering the resulting SNR, AG-VPCM is superior to A-VPCM by 0.42 dB in Test.mode and by 0.02 dB in Train.mode. Therefore, the cycle to cycle dependency is not effectively utilized by AG-VPCM. The reason for that is again due to the existence of the silence speech and unvoiced sounds.

Considering the resulting SEGSNR, AG-VPCM is superior to A-VPCM by 4.03 dB in Test.mode and by 1.62 dB in Train.mode. Therefore, the dependencies among the normalized speech cycles of different voiced sounds are not effectively utilized by AG-VPCM (only 1.62 dB increase in SEGSNR is achieved in Train.mode). The reason for that is the degree of dependencies. As the r(1) of Class 2 ranges from 0.94 down to 0.7, the waveform of the speech cycles of all the voiced sounds that belong to this class varies a lot more than that of those belong to Class 1. The normalized speech cycle of different voiced sounds are less dependent on each other than that of those belong to Class 1.

(iii) Class 3

The 80240 samples training sequence is undoubtly too short for the codebook design of both AG-VPCM and A-VPCM. The resulting SNR of AG-VPCM and of A-VPCM in Test.mode are different from those of AG-VPCM and of A-VPCM in Train.mode by 2.2 dB and by 7.47 dB, respectively. In addition, some of the codewords designed for A-VPCM are found to be the "copies" of the input speech vectors, thereby "zero distortion" results when the input vectors are reconstructed by "themselves". The existence of these codewords is mainly

the reason why A-VPCM results in a SNR of 9.93 dB in Train.mode and of only 2.46 dB in Test.mode. Therefore, in considering the resulting SNR and SEGSNR, the results in Test.mode are more trustworthy than those in Train.mode.

From the results in Test.mode, AG-VPCM is superior to A-VPCM by 2.76 dB in SNR and by 3.15 dB in SEGSNR. Therefore, the dependencies between the normalized voiced and unvoiced sounds are effectively utilized by AG-VPCM.

(iv) The overall system

Considering the overall system, AG-VPCM is superior to A-VPCM by 0.76 dB in SNR and by 3.75 dB in SEGSNR. Therefore, based on the resulting SEGSNR, it is assumed that AG-VPCM provides better quality output speech than A-VPCM. However, the SEGSNR improvement is less than that achieved in the non-adaptive system. Approximately a 5 dB increase in SEGSNR is achieved by G-VPCM over VPCM.

As a conclusion, although AG-VPCM is better in dependencies utilization than G-VPCM, it is not as efficient in SEGSNR improvement as G-VPCM. In the following section, an inter-coder comparison, based on the performance and the transmission bit rate, is given. After the comparison, it will be clear that AG-VPCM is the best out of all the discussed coding schemes.

3.9 Comparison between different VPCM-based systems

Throughout this section, only the performance and the transmission bit rate of each VPCM-based system are of

interest. The computation complexity of each VPCM-based system will be discussed in the next chapter. It should be emphasized that only the resulting SNR and SEGSNR obtained in Test.mode are used for inter-coder comparison. For an arbitrary speaker, these results correlate better to the "real performance" of each VPCM-based system. These results together with the transmission bit rates of the VPCM-based systems are shown in Table 3.8.

From Table 3.8, AG-VPCM results in the highest SNR and SEGSNR but it also requires the highest transmission bit rate. Also, the systems that require a higher transmission bit rate result in higher SNR and SEGSNR. Therefore, it is almost impossible to conclude, based on the SNR, the SEGSNR and the transmission bit rate, which VPCM-based system is better than the others.

However, if we limit ourselves in considering only the SEGSNR and the transmission bit rate, the best VPCM-based system can be found easily. Introduce a performance measure, SEGSNR/kb/s, defined as

SEGSNR/kb/s(dB) = SEGSNR(dB)/ transmission bit rate (kb/s)

That is, the SEGSNR/kb/s is the normalized SEGSNR of a coding system with respect to the corresponding transmission bit rate. It represents, on the average, the SEGSNR for every 1kb/s transmission bit rate of a coding system. Therefore, it is very useful in comparing different coding systems with different transmission bit rates. However, the

	SNR (dB)	SEGSNR(dB)	Transmission bit rate (kb/s)
VPCM	12.24	9.96	8
G-VPCM	12.68	14.55	9.6
A-VPCM	12.56	11.40	8.2
AG-VPCM	13.32	15.15	9.8

Table 3.8 Performances and transmission bit rates of theVPCM-based systems

	SEGSNR/kb/s (dB/kb/s)
VPCM	1.25
G-VPCM	1.52
A-VPCM	1.39
AG-VPCM	1.55

Table 3.9 The SEGSNR/kb/s of theVPCM-based systems

SEGSNR/kb/s is not proportional to the transmission bit rate, thereby it is not applicable for the estimation of the SEGSNR of a coding scheme at any transmission bit rate. Finally, it must be kept in mind that the SEGSNR/kb/s is defined for comparing the VPCM-based systems. It is not a proven performance measure and it does not exist in the literature. The SEGSNR/kb/s of the VPCM-based systems are shown in Table 3.9

From Table 3.9, it is obvious that AG-VPCM, which results in the highest SEGSNR/kb/s, is the best. Note that G-VPCM is inferior to AG-VPCM by only 0.03 dB/kb/s and by 0.6 dB in SEGSNR. Therefore, it is still not clear whether AG-VPCM or G-VPCM should be chosen as the desired speech coder. This is discussed in the next section.

3.10 The desired speech coder

From the previous section, it is obvious that either AG-VPCM or G-VPCM is desired. In this section, it is assumed that both AG-VPCM and G-VPCM are implementable. Hence, the critical parameters for judgement are the SNR, the SEGSNR and the SEGSNR/kb/s. Note further that the transmission bit rate of AG-VPCM is only 200 bits/s higher than that of G-VPCM. Whether AG-VPCM or G-VPCM is desired can be judged using only the SNR and the SEGSNR.

As AG-VPCM is superior to G-VPCM only by 0.64dB in SNR and 0.6 dB in SEGSNR, it is helpful to take into account the reconstructed waveforms for performance judgement. A 300ms original speech waveform, the corresponding speech waveforms

reconstructed by G-VPCM and AG-VPCM are shown in Fig. 3.9, Fig. 3.10 and Fig. 3.11, respectively. From Fig. 3.9-3.11, the waveforms which are reconstructed by AG-VPCM and G-VPCM are both similar to the original waveform, except for the discontinuity in Fig. 3.10. As suggested in [2], a discontinuity in the reconstructed speech waveform may be heard as roughness, AG-VPCM is concluded to provide better quality of output speech.

Finally, although AG-VPCM provides the best quality of output speech, its output speech is not necessarily of communication quality. Fortunately, because of the 8-dimensional VPCM of 1 bit/sample in [15], the subjective quality of the output speech of AG-VPCM can be judged without a listening test. In [15], the 8-dimensional VPCM was tested, using real-time computer simulation, by a speech sequence of about 200 seconds. The test results were objectively a SNR of 13.6 dB and, subjectively by listening test, an intelligible and naturally sounding reconstructed speech. Since the SNR of AG-VPCM is 13.32 dB, which is approximately equal to 13.6 dB, AG-VPCM can be expected to provide intelligible and naturally sounding speech output. Thus, the output speech of AG-VPCM is certainly not of synthetic quality. As a conclusion, the output speech of AG-VPCM is of communication quality.

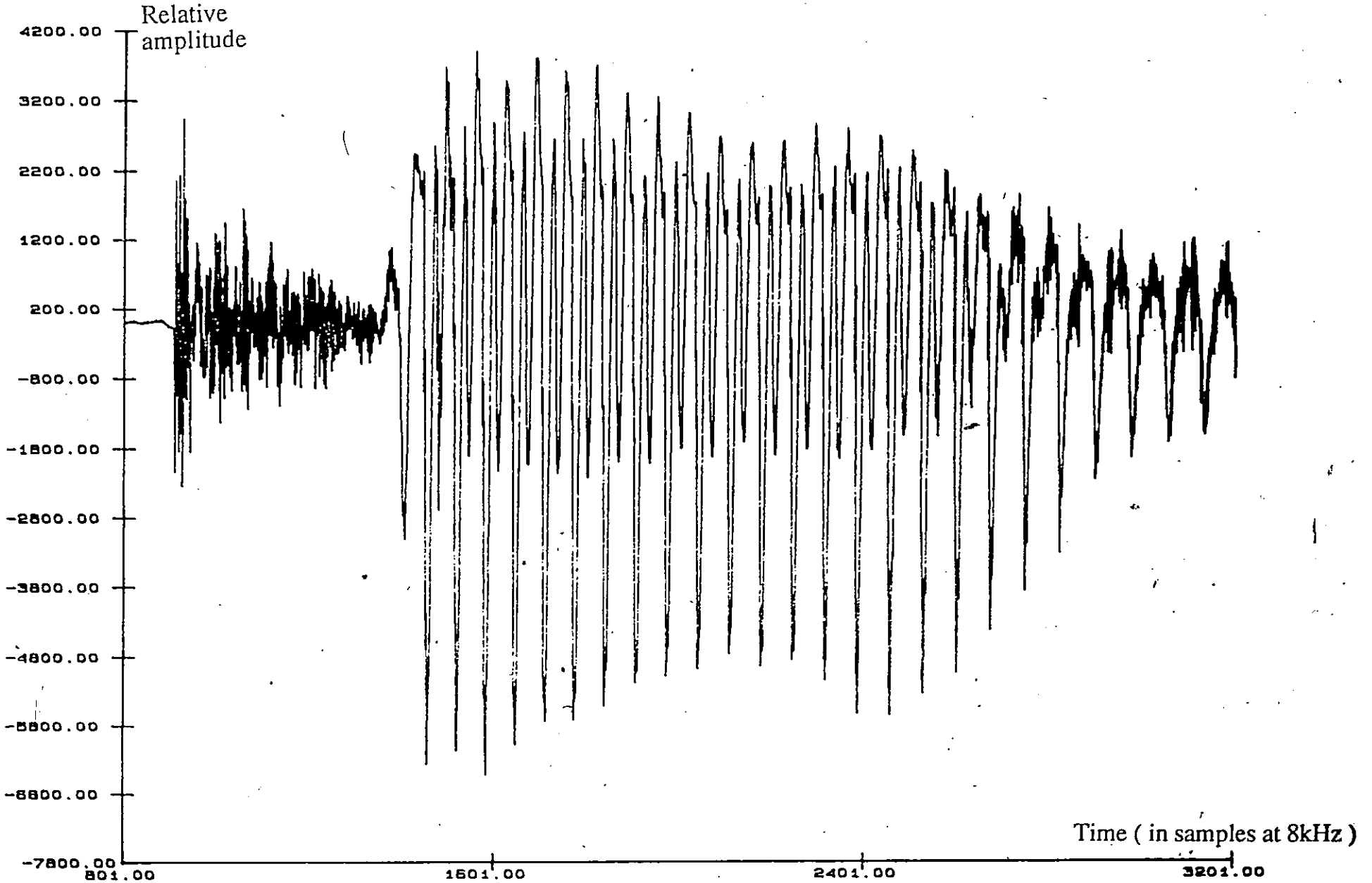


Fig. 3.9 Original speech waveform of [JT.24]

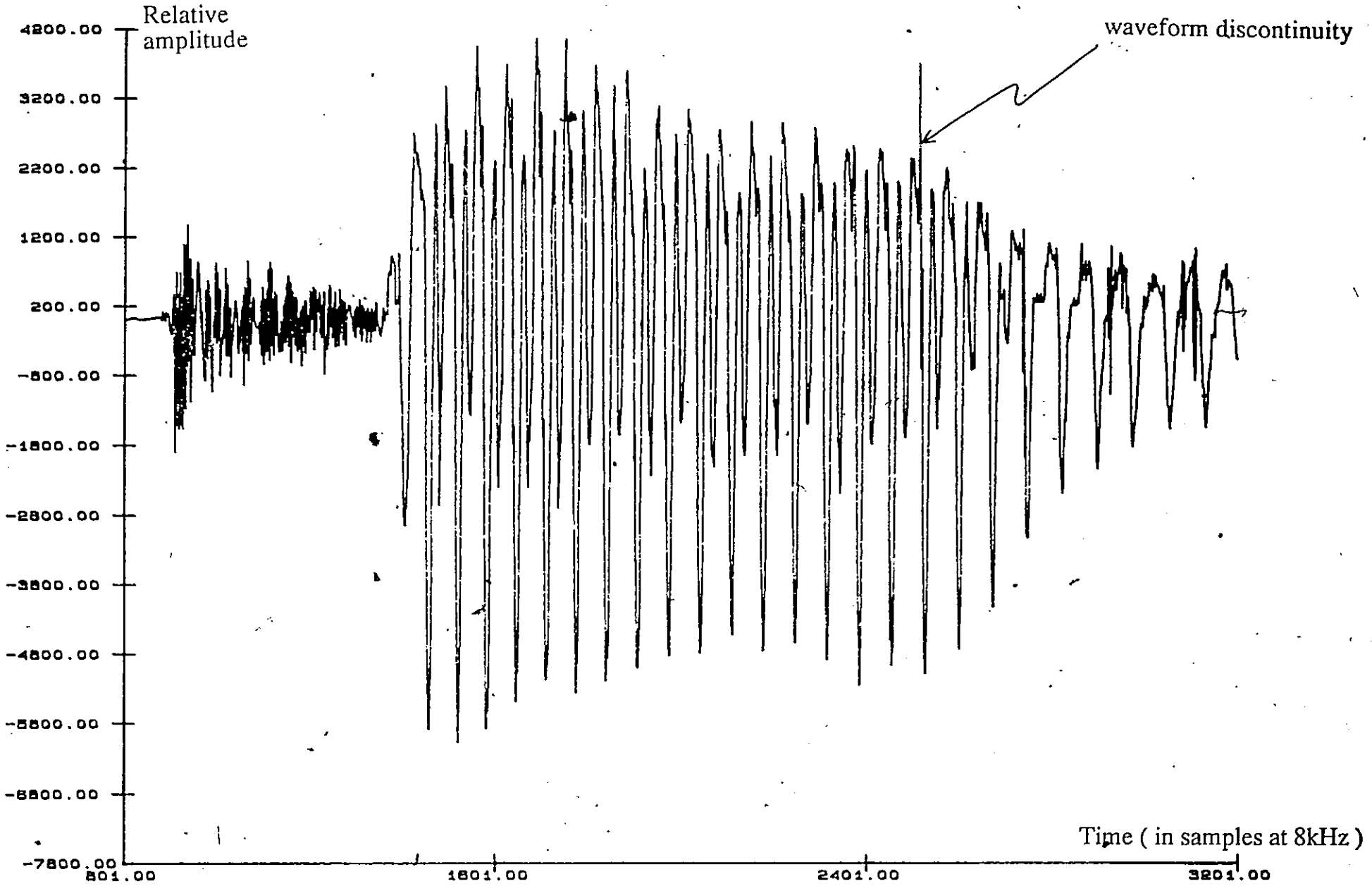


Fig. 3.10 The speech waveform of [JT.24] reconstructed by G-VPCM

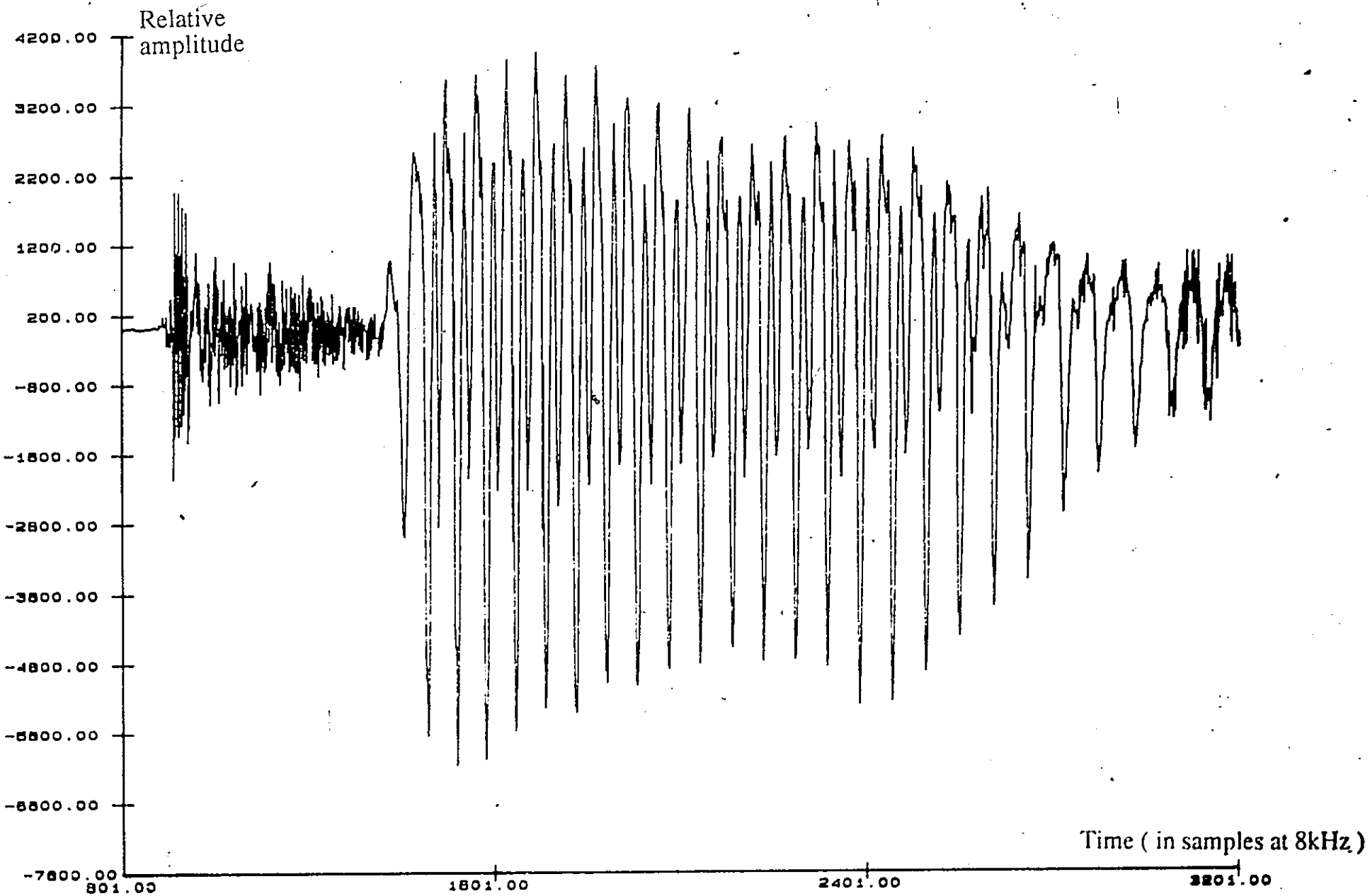


Fig. 3.11 The speech waveform of [JT.24] reconstructed by AG-VPCM

3.11 Conclusion

After a lengthy discussion on the development of different extensions of VPCM, the normalization scheme in G-VPCM is found to be very efficient in SEGSNR improvement. However, G-VPCM shows a discontinuity in its reconstructed speech waveform. The modified fixed adaptive technique when applied in G-VPCM is found to be not significant in SEGSNR improvement, but managed to solve the problem of the waveform discontinuity. As a result, AG-VPCM is determined as the medium-band speech coder that provides communication quality.

CHAPTER 4

COMPUTATION COMPLEXITY OF THE VPCM-BASED SYSTEMS

4.1 Introduction

Given the structure of all VPCM-based systems, it is possible to discuss in detail their computation complexities and real-time realizations. The computation complexities are measured in terms of the number of mathematical operations such as multiplications and additions. As expected, all extensions of VPCM are shown to have a minimum increase in the computation complexity. Using the VLSI Pattern Matching Chip (PMC) [25], the selected speech coding scheme AG-VPCM is shown to be real-time implementable. In addition, this chapter also provides the information on the real-time implementation of AG-VPCM using the fast search algorithm [26].

As discussed in chapter 2, the encoding process of the full search vector quantizer is far more complicated than the decoding process. Therefore, only the computation complexity of the encoding process is necessarily considered for the hardware realization of the full search vector quantizer. Hence, only the computation complexities of the encoding process of each extension of VPCM are discussed in this chapter. Also, the effect of the fast search algorithm [26] on reducing the computations of the encoding process of VPCM is discussed.

The computation complexities of all the VPCM-based systems are given in section 4.2. In section 4.3, the AG-

VPCM encoder is shown to be real-time implementable and a hardware architecture is proposed. The hardware architecture is suitable for the realization of the AG-VPCM encoder using either the PMC or the fast search algorithm.

4.2 Computation complexities of all the VPCM-based systems

Let mult. denotes multiplications

add. denotes additions

sub. denotes subtractions

comp. denotes comparisons

div. denotes divisions

For each VPCM-based system, the computation complexity is expressed in terms of the above notations and the corresponding system parameters. Based upon these expressions, a comparison between the different VPCM-based systems with the parameters of interest is given.

4.2.1 Computation complexity of VPCM

Let $x(i)$, $i=1,\ldots,k$ be an arbitrary input vector, and $y_n(i)$, $i=1,\ldots,k$; $n=1,\ldots,N$ be the designed codewords

Then, based upon the unweighted squared error distortion measure, the computations for the encoding process of VPCM are the following:

(i) for all N codewords, compute the error vectors, $e_n(i)$, $i=1,\ldots,k$; $n=1,\ldots,N$ as

$e_n(i) = x(i) - y_n(i)$ which requires Nk sub.

(ii) compute the squared errors, SE(n), n=1,...,N for each error vector as

$$SE(n) = \sum_{i=1}^{k} [e_n(i)]^2$$ which requires Nk mult. and, N(k-1) add.

(iii) select the minimum distortion codeword for x(i), i=1,...,k through the following:

min(SE(n)), for all n, which requires (N-1) comp.

Therefore, the computation complexity of VPCM is:

(VPCM computations)=(mult., add., sub., comp.)

=(Nk, N(k-1), Nk, N-1)

On a per sample basis:

(VPCM computations/sample)=(mult., add., sub., comp.)

=(N, N(k-1)/k, N, (N-1)/k)

4.2.2 Computation complexity of G-VPCM

From chapter 3 (section 3.5), the computation complexity of G-VPCM equals that of VPCM plus the computations needed for the normalization scheme.

Let L = segment-length in samples and L>k

s(n), n=1,...,L be an arbitrary speech segment

$s_{norm}(n)$, n=1,...,L be the normalized s(n)

Then, the computation complexity of the normalization scheme is the following:

(i) Find the maximum sample magnitude, a, of s(n) as

a = max(s(n)), For all n, which requires (L-1) comp.

(ii) normalizes s(n) as

$$s_{norm}(n) = \frac{s(n)}{a}$$ which requires L div.

Therefore, the number of computations per sample for the N-level k-dimensional G-VPCM is:

(G-VPCM computations/sample)

=(mult., add., sub., comp., div.)

=(0, 0, 0, L-1, L)/L +(VPCM computations/sample)

=(0, 0, 0, 1-1/L, 1) +(N, N(k-1)/k, N, (N-1)/k, 0)

=(N, N(k-1)/k, N, 1-1/L +(N-1)/k, 1)

4.2.3 Computation complexity of A-VPCM

From chapter 3 (section 3.7.2), the computation complexity of A-VPCM is the number of VPCM computations plus the number of computations for the classifier.

Then, the computation complexity of the classifier is the following:

(i) compute the zero delay autocorrelation coefficient, R(0), as

$$R(0) = \sum_{n=1}^{L} s(n)s(n)$$ which requires L mult. and, (L-1) add.

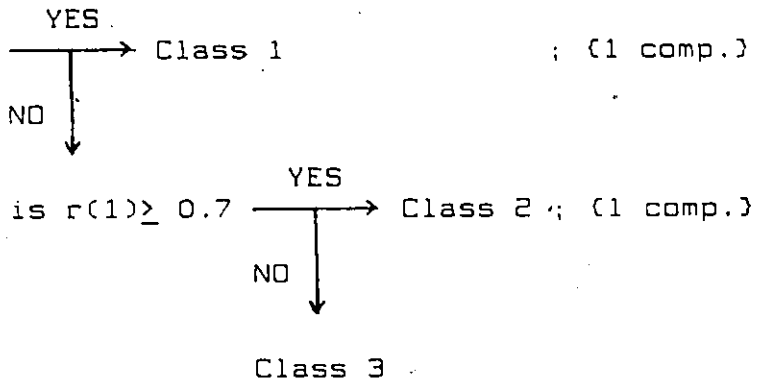
(ii) compute the unit delay autocorrelation coefficient, R(1), as

$$R(1) = \sum_{n=1}^{L-1} s(n)s(n+1)$$ which requires (L-1) mult. and, (L-2) add.

(iii) compute the normalized unit delay autocorrelation coefficient, r(1), as

$$r(1) = \frac{R(1)}{R(0)}$$ which requires 1 div.

(iv) select the appropriate class for s(n) as



Therefore, the number of computations per sample for the N-level k-dimensional A-VPCM is:

(A-VPCM computations/sample)

=(mult., add., sub., comp., div.)

=(2L-1, 2L-3, 0, 2, 1)/L +(VPCM computations/sample)

=(2-1/L, 2-3/L, 0, 2/L, 1/L)

+(N, N(k-1)/k, N, (N-1)/k, 0)

=(N+2-1/L, N(k-1)/k +2-3/L, N, 2/L +(N-1)/k, 1/L)

4.2.4 Computation complexity of AG-VPCM

The computation complexity of AG-VPCM is equal to that of VPCM plus the computations for the normalization scheme and the classifier.

From section 4.2.2, the number of computations per sample for the normalization scheme is:

(nor. computations/sample)=(mult., add., sub., comp., div.)

=(0, 0, 0, 1-1/L, 1)

Further, from section 4.2.3, the number of computations per sample for the classifier is:

(class. computations/sample)

=(mult., add., sub., comp., div.)

=(2-1/L, 2-3/L, 0, 2/L, 1/L)

Therefore, the number of computations per sample for the N-level k-dimensional AG-VPCM is:

(AG-VPCM computations/sample)

=(nor. computations/sample)

+(class. computations/sample)

+(VPCM computations/sample)

=(mult., add., sub., comp., div.)

=(0, 0, 0, 1-1/L, 1) +(2-1/L, 2-3/L, 0, 2/L, 1/L)

+(N, N(k-1)/k, N, (N-1)/k, 0)

=(N+2-1/L, N(k-1)/k +2-3/L, N, (N-1)/k +1+1/L, 1+1/L)

4.2.5 Comparison between different VPCM-based systems

From chapter 3, the system parameters of interest are:

vector dimension k=8

codebook size N=256

sample frequency f_s=8kHz

segment-length L=10ms =80 samples

Based upon the derived computational expressions, the number of computations/sample for each VPCM-based system with respect to the above parameters are shown in Table 4.1.

per sample computations	mult.	add.	sub.	comp.	div.
VPCM	256	224	256	31.875	0
G-VPCM	256	224	256	32.8625	1
A-VPCM	257.9875	225.9625	256	31.9	0.0125
AG-VPCM	257.9875	225.9625	256	32.8875	1.0125

Table 4.1 Computation complexities of the VPCM-based systems (systems parameters: k=8 ; N=256 ; L=80 samples)

per sample computations		mult. and div.	add. and sub.	comp.
VPCM	full search method	1024	1920	127.875
fast search algorithms	Projection Method	0.75	0.75	225
	Hypercube Method	50	100	240
	Minimax Method	3.5	1026	1154

Table 4.2 Comparison of computation complexity. After [26] (system parameters: k=8 ; N=1024)

Referring to Table 4.1, it is obvious that the computational requirements of all the VPCM-based systems are approximately equal to each other. Hence, the extra computation required by any of the extensions of VPCM is negligible when compared to that of VPCM. In other words, if the VPCM of interest is real-time realizable, then all the other VPCM-based systems are also real-time realizable. This is further discussed in the following section.

4.3 Real-time realization of AG-VPCM

As AG-VPCM is the most complex extension of VPCM, a real-time implemented AG-VPCM implies the real-time realizations of all the other VPCM-based systems. Also, AG-VPCM is the selected speech coding scheme. Therefore, only the hardware realization of AG-VPCM will be discussed.

From Table 4.1, the main obstacle of implementing AG-VPCM is the VPCM computations. Two attempts have been made to achieve the VPCM computations in real time. The first is to use the PMC [25]. The second attempt makes use of the fast search algorithms [26].

4.3.1 Real-time realization using the PMC

The PMC is a 4-micron NMOS VLSI chip with a clock frequency of 4MHz. Its architecture allows a selectable vector dimension, k, of size 2,4,6 or 8. Also, the PMC is capable of doing a squared difference calculation for a single vector component within a single clock cycle, T_{VC}, of 250ns.

If the PMC is applied to realize the VPCM using sequential processing of the vector components, the processing time of the PMC, T_p, is:

$T_p = T_{vc} \cdot k \cdot N$ seconds/input vector

or $T_p = T_{vc} \cdot N$ seconds/sample

For the system of interest, N=256 and f_s=8kHz,

$T_p = 250 \times 10^{-9} \times 256 = 64\ \mu s$/sample

and, the allowable operating time, T_a, is

$T_a = 1/f_s = 125\ \mu s$/sample

Hence, there are 61 μs/sample allowable for the AG-VPCM to normalize and classify the speech segment.

From section 4.2.5, the computations of the normalization scheme and the classifier with the system parameters of interest are:

(nor. computations/sample)=(mult., add., sub., comp., div.)

=(0, 0, 0, 0.9875, 1)

(class. computations/sample)=(mult., add., sub., comp.,div.)

=(1.9875,1.9625,0, 0.025,0.0125)

Thus, the extra computations of the AG-VPCM compared to the VPCM computations are:

(extra computations/sample)

=(nor. computations/sample)

+(class. computations/sample)

=(mult., add., sub., comp., div.)

=(1.9875, 1.9625, 0, 1.0125, 1.0125)

If one can build hardware completing the above computations within 61μs, the AG-VPCM is real-time

implemented. To build such hardware is certainly not a difficult task. For instance, the Texas Instruments TMS320 chip is capable of completing each of the required mathematical operations in 200ns. Using the TMS320, one can do all the required calculations in

$$(1.9875+1.9625+1.0125+1.0125)\times 200\ ns = 1.195\mu s$$

which is about 59μs faster than required. Thus, using the PMC together with the TMS320, the AG-VPCM is shown to be real-time implementable.

4.3.2 Real-time realization using the fast search algorithm

From section 4.2.1, the computation complexity of VPCM is:

(VPCM computations)=(mult., add., sub., comp., div.)

$$=(Nk,\ N(k-1),\ Nk,\ N-1,\ 0)$$

Note that the number of each kind of mathematical operation required is increasing with increasing N and/or k. Therefore, the number of VPCM computations can be reduced by reducing the N and/or k. This is addressed by the fast search algorithms [26].

There are three different fast search algorithms developed in [26]. Each of these algorithms can be divided into two stages. The first stage of each of the three algorithms sucessfully selects a subset of the N codewords in the codebook (the subset consists of N_S codewords with N_S is much smaller than N) for the search of the nearest neighbor of a given input vector. Hence, the number of VPCM computations is substantially reduced as the given N is reduced to

$N_S << N$. However, the size of the selected subset, N_S, is a random variable depending on the particular input vector to be encoded. Therefore, the amount of computational reduction in VPCM is varying with respect to the input vectors.

The second stage of all the three algorithms searches for all the N_S codewords and selects the nearest neighbor of the given input vector using the "partial distance" technique [26]. This technique consists of computing the m^{th} partial distance or the m-dimensional distance (m is smaller than the given dimension k), the sum of the squares of the first m components of the difference between the given input vector and a codeword. During the test of a new codeword, if the m-dimensional distance exceeds the previous minimum k-dimensional distance, the codeword is rejected without completing the calculation of the distortion (the k-dimensional distance) between the new codeword and the given input vector. It has been found [26] that the partial distance technique reduces the number of multiply and add operations to an average of 25% of the VPCM computations, while increasing the number of comparsions by a factor of about k/4.

In order to show the performance of the fast search algorithms, the results in [26] are shown in Table 4.2. These results were the average number of computations obtained from the 14720 8-dimensional test vectors.

According to these results, the number of VPCM computations are substantially reduced. Thus, for a given

processor capability, a much faster nearest neighbor search or a larger codebook size can be realized using the fast search algorithms. For instance, if the full search method is implemented using the TMS320, the average operating time of the TMS320 will be

(1024+1920+127.875)X200 ns =614.375 μs/sample

If the TMS320 is used to implement the Hypercube Method [26], then the average operating time of the TMS320 is

(50+100+240)X200 ns =78 μs/sample

which is about 8 times faster than the full search method. Note further that for an 8kHz sampling frequency, the full search method is not possibly real-time implemented using a single TMS320.

However, the fast search algorithms are not ready for real-time applications. The objective of using the fast search algorithm is to reduce the computational requirement of the VPCM, thereby allowing existing hardware to do the VPCM computations in real time. What is required is the maximum computation complexity rather than the reported [26] average computation complexity of the nearest neighbor search.

To make the fast search algorithm applicable in real time, the author believes that it is helpful to start with the following steps:

(i) Find the maximum possible size of the selected subset of the N codewords. That is, find the maximum value of N_s. It should be emphasized that due to the different

nature of the three different algorithms, the max(N_s) changes from one algorithm to another [26]. Hence, care must be taken in the selection of the max(N_s).

(ii) Find the maximum computation complexity of the nearest neighbor search resulting from using the partial distance technique, with respect to the input vectors.

Based upon the above steps, one can estimate the maximum computation complexity of the nearest neighbor search resulting from using any of the fast search algorithm. Subsequently, one can proceed to select the appropriate hardware.

4.3.3 Hardware architecture for AG-VPCM

The hardware architecture for AG-VPCM is an extension of that developed in [24]. The architecture of [24] supports a real-time realization of vector quantization. Based upon this architecture, a VLSI Codebook-Search Processor (CSP) was developed [25]. This CSP was found to be useful for the real-time realizations of VPCM and AVPC. Indeed, this CSP is useful for any vector quantizer which makes use of the squared error distortion measure.

The proposed hardware architecture for the encoder of the AG-VPCM is shown in Fig. 4.1. This architecture can be realized using either the PMC or the fast search algorithm to implement the AG-VPCM search processor. The block diagrams of the AG-VPCM implementation using the PMC and the fast search algorithm are shown in Fig. 4.2 and Fig. 4.3,

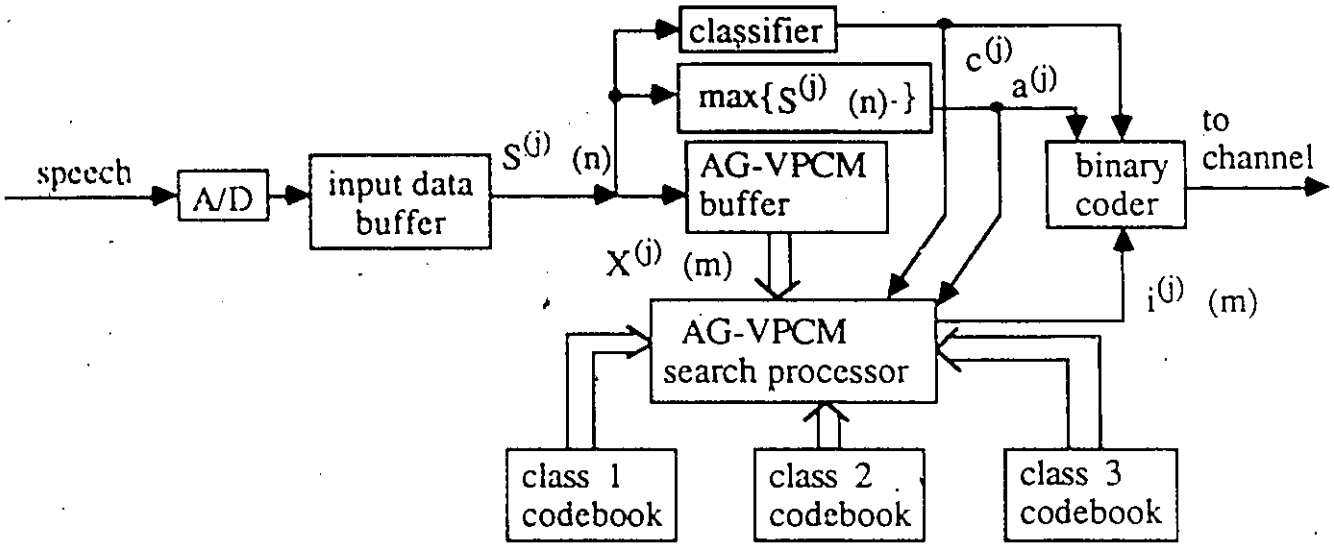


Fig.4.1 Block diagram of the hardware architecture for the encoder of AG-VPCM

Note :

- $S^{(j)}(n)$ = the n^{th} sample of the j^{th} speech segment ; n=1,...,L
- L = the length of the speech segments
- $c^{(j)}$ = the class identification of the j^{th} speech segment
- $a^{(j)}$ = the maximum sample magnitude of the j^{th} speech segment
- $Y^{(j)}(m)$ = the m^{th} vector of the j^{th} normalizated speech segment ; m=1,...,M
- $M = \frac{L}{k}$ where k is the vector dimension
- $i^{(j)}(m)$ = the index of the nearest neighbor of $Y^{(j)}(m)$

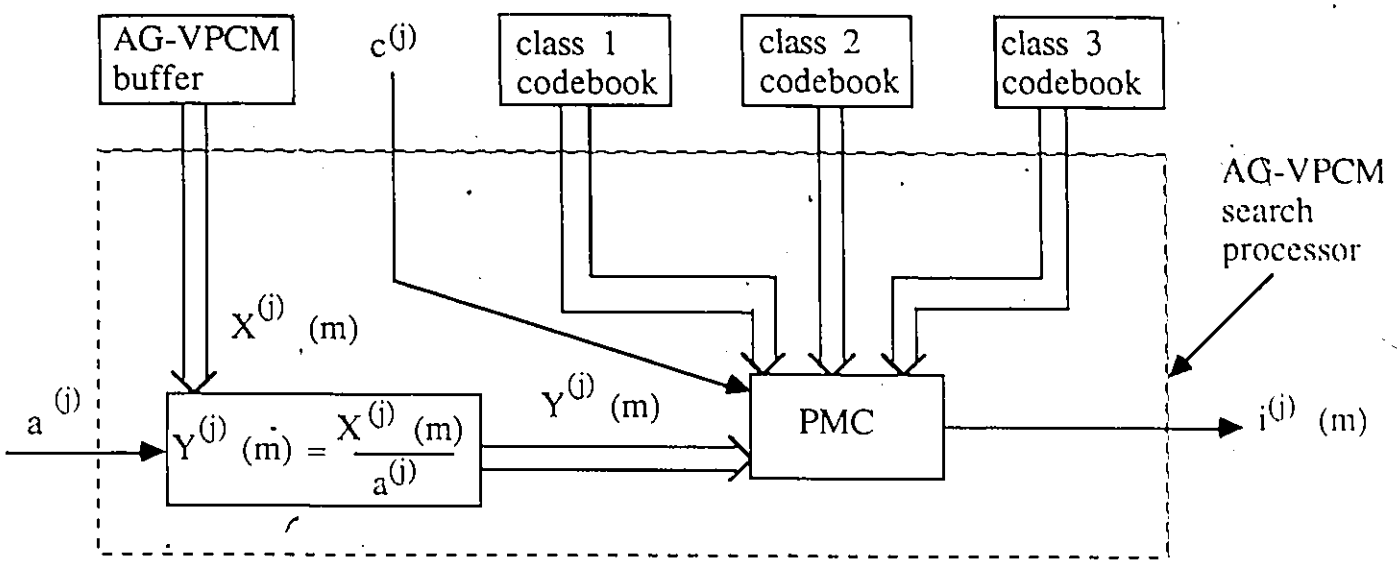


Fig.4.2 Block diagram of the AG-VPCM search processor (implemented using the Pattern Matching Chip, PMC)

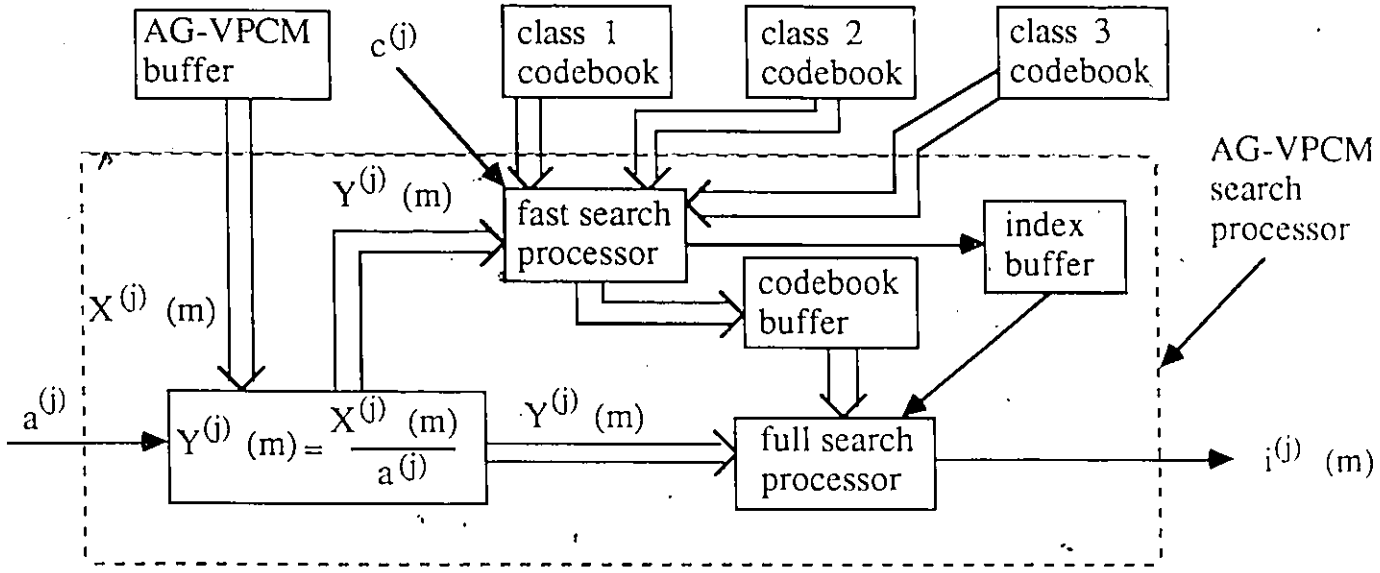


Fig.4.3 Block diagram of the AG-VPCM search processor (implemented using the fast search algorithm)

respectively. Based upon the proposed hardware architecture, the encoding process of the AG-VPCM consists of the following steps:

(1) Convert the analog speech into digital form with a A/D converter.

(2) Collect every consecutive group of L samples as a speech segment by the input data buffer, and load the current speech segment into the AG-VPCM buffer.

(3) Find the normalization factor, $a^{(j)}$, of the current speech segment, $s^{(j)}(n)$, and classify $s^{(j)}(n)$ into one of the three classes.

(4) After the class identification, $c^{(j)}$, is obtained, the AG-VPCM search processor is able to select the proper codebook for $s^{(j)}(n)$.

(5) The AG-VPCM search processor takes sequences of k samples (vector) from the AG-VPCM buffer and normalizes each sample.

Goto step (9) if the AG-VPCM search processor is implemented using the PMC.

(6) After the current vector, $X^{(j)}(m)$, is normalized, the fast search processor, which realizes the first stage of either one of the three fast search algorithms [26], is acknowledged to perform the selection of an appropriate subset of the currently selected codebook for the normalized $X^{(j)}(m)$, $Y^{(j)}(m)$.

(7) Every codeword and its index, which were selected by the fast saarch processor, are then loaded into the codebook buffer and the index buffer, respectively.

(8) The full search processor, then, searches through the codebook buffer to find the minimum distortion codeword. Note that the full search processor is expected to be realized using the partial distance technique.

Goto step (10)

(9) After the current vector, $X^{(j)}(m)$, is normalized, the PMC searches through the currently selected codebook to find the minimum distortion codeword.

(10) After the nearest neighbors of every vector from $s^{(j)}(n)$ are found, the indexes of the nearest neighbors, $i^{(j)}(m)$ for all m, together with $c^{(j)}$ and $a^{(j)}$ are binary coded for transmission.

(11) The binary coder may employ certain error detection codes or even error correction codes when the channel noise conditions require it.

4.3.4 Comments on the proposed hardware architecture

The hardware architecture described previously is not necessarily the final architecture of the real-time AG-VPCM. This proposal provides just enough information for the implementation of AG-VPCM. It is included to show that the real-time realization of AG-VPCM is not a difficult task.

Based upon the architecture of [24], a two codebook VPCM (using separate codebooks for voiced and unvoiced sounds) was sucessfully realized. The two codebook VPCM is more complicated than AG-VPCM. The voiced/unvoiced/silence (VUS) classifier [22] for the two codebook VPCM is much more complicated than the classifier together with the

normalization scheme in AG-VPCM. Hence, one can realize the AG-VPCM based upon the proposed architecture.

Finally, note that if a single codebook is used for AG-VPCM, the hardware architecture can be used for G-VPCM without any extension.

CHAPTER 5

CONCLUSION

Throughout this thesis work, it was found that both AG-VPCM and G-VPCM resulted in significant SEGSNR improvement over VPCM. However, the output speech of G-VPCM resulted in waveform discontinuity. AG-VPCM was, then, selected to be the desired speech coder.

The 9.8 kb/s AG-VPCM was found to satisfy all the projected requirements. Its output speech is "intelligible and natural sounding" and, it is real-time realizable.

As no listening tests were conducted for AG-VPCM and G-VPCM, the absolute subjective quality of the output speech of the 9.8 kb/s AG-VPCM is not yet known. Referring to the SEGSNR measure and the reconstructed speech waveform (Fig. 3.11), the author believes that the 9.8 kb/s AG-VPCM does provide good communication quality of output speech.

The 9.8 kb/s AG-VPCM can be implemented by combining the PMC with general-purpose signal processors such as the TMS320. Although the fast search algorithms [26] are not ready to be used in real time, they are potentially good candidates for the real-time realization of AG-VPCM. An AG-VPCM with a large codebook size (N>256 and/or k>8) can be realized by combining the fast search algorithm and the PMC with general-purpose DSP chips. Future research on the real-time realization of the 9.8 kb/s AG-VPCM using the fast search algorithm will be very interesting and challenging.

APPENDIX A

DIGITAL MODEL FOR VOICED SPEECH PRODUCTION

The basic model, detailed by L. R. Rabiner and R. W. Schafer [30], for speech production is the following:

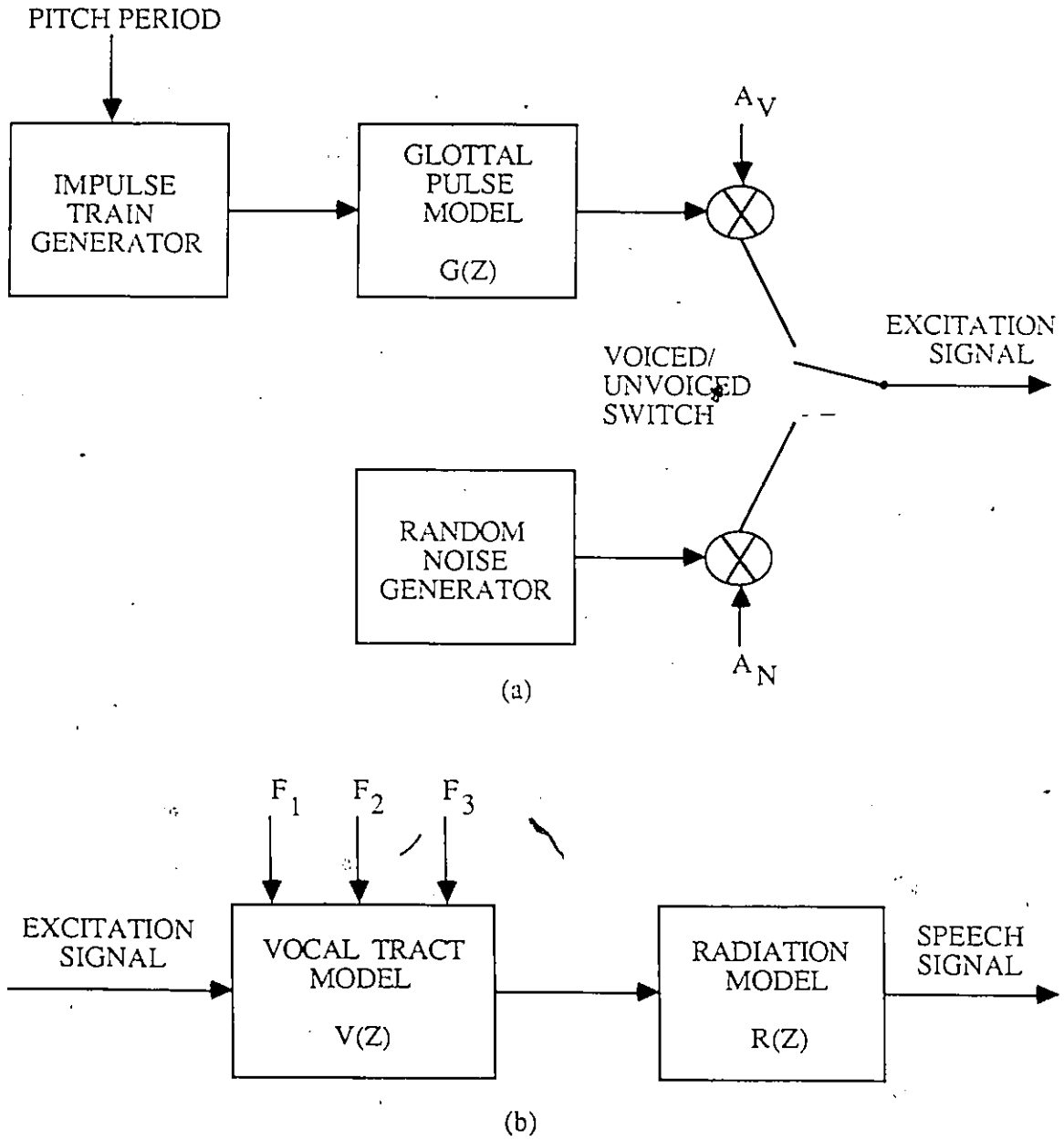


Fig.A.1 (a) Generation of the excitation signal;
(b) Terminal model for speech production

As shown in Fig. A.1(a), this model makes use of different excitation sources for voiced and unvoiced sound production. For the voiced sound production, the excitation source is generated by the impulse train generator, the glottal pulse model and the amplitude control, A_V. For the unvoiced sound production, the excitation source is the random noise generator and the amplitude control, A_N.

According to this model (Fig. A.1(a),(b)), the speech production is based on the "quasi-stationarity" assumption. That is, the model assumes the general properties of the excitation and vocal tract remain fixed for periods of 10-20ms, thus its parameters are assumed to be constant over time intervals of 10-20ms long. Therefore, the model works very well for sounds such as vowels with parameters that change very slowly with time, but is not adequate for voiced/unvoiced transitions with parameters changing suddenly at some points in time.

In addition, the lack of provision of zeros (the vocal tract model is an all pole model) as required theoretically for nasals and fricatives leads to "strange" sounds produced for nasals and voiced fricatives. Nevertheless, for the purpose of voiced sound production only, the model works very well, and the random noise generator and the A_N are not necessary. Such a simplified model is the one shown in Fig. 3.5. The details of this model follow. The method for fixed pitch voiced sound production is also discussed subsequently.

Glottal pulse model: G(Z)

The objective of the glottal pulse model together with the impulse train generator is to create the excitation signal (the quasi-periodic pulse waveform) for the voiced sound production. The impulse train generator produces a sequence of unit impulses which are spaced by the desired pitch period. This signal is, then, used to excite the glottal pulse model with impulse response g(n) having the desired glottal wave shape. The choice of the glottal wave shape is not critical, as long as the selected g(n) exhibits a lowpass filtering effect in the frequency domain [30]. A commonly used approximation to the glottal pulse is the following:

$$g(n) = \begin{cases} na^n, & n \geq 0 \\ 0, & n < 0 \end{cases} \qquad (A1)$$

where "a" is chosen subject to the condition of

$$20 \log|G(e^{j0})| - 20 \log|G(e^{j\pi})| = 60 \text{ dB}$$

or, equivalently the condition of

$$\left|\frac{G(e^{j0})}{G(e^{j\pi})}\right| = 1000 \qquad (A2)$$

To calculate the G(Z), let:

$$h(n) = \begin{cases} a^n, & n \geq 0 \\ 0, & n < 0 \end{cases} \qquad (A3)$$

Then, the Z-Transform of h(n) is

$$H(Z) = \sum_{n=0}^{\infty} a^n Z^{-n}$$

$$= \frac{1}{1 - aZ^{-1}} = \frac{Z}{Z - a} \qquad (A4)$$

Therefore,

$$G(Z) = -Z \frac{dH(Z)}{dZ} = -Z \frac{(Z - a) - Z}{(Z - a)^2}$$

$$= \frac{aZ}{(Z - a)^2} = \frac{aZ^{-1}}{1 - 2aZ^{-1} + a^2Z^{-2}} \qquad (A5)$$

Subject to (A2), the parameter a is obtained as follows:

From A(5),

$$G(e^{j\omega}) = \frac{ae^{j\omega}}{(e^{j\omega} - a)^2} \longrightarrow G(e^{j0}) = \frac{a}{(1 - a)^2} ;$$

$$G(e^{j\pi}) = \frac{-a}{(-1 - a)^2}$$

$$= \frac{-a}{(1 + a)^2}$$

Thus,

$$\left| \frac{G(e^{j0})}{G(e^{j\pi})} \right| = 1000 = \frac{a/(1 - a)^2}{a/(1 + a)^2} = \frac{(1 + a)^2}{(1 - a)^2}$$

Hence,

$$1000(1 - 2a + a^2) = 1 + 2a + a^2$$

or $$999a^2 - 2002a + 999 = 0$$

By solving the above quadratic equation, a is given by

$$a = 1.065310864 \text{ or } 0.938693139$$

For a stable G(Z), the desired parameter a is 0.93869139. As the amplitude of G(Z) is controlled by the A_0, it is better

to normalize G(Z). Since there is only one double pole at Z=a and a is real, $G(e^{j\omega})$ has a distinct maximum at Z=a (zero frequency) and the amplitude of $G(e^{j\omega})$ has a roll-off of 40 dB/decade at Z=a (this is the required lowpass filtering effect). Therefore, to normalize G(Z) at Z=a is sufficient. Finally, as the desired value of a is about 1, to normalize G(Z) at Z=1 is sufficient. Hence, the final expression of G(Z) is

$$G(Z) = \frac{(1 - a)^2 Z^{-1}}{1 - 2aZ^{-1} + a^2 Z^{-2}} \quad \text{with } a=0.93869139 \qquad (A6)$$

Note that at Z=1 (zero frequency), G(1) = 1.

Vocal tract model: V(Z)

Since the effect of the vocal tract is to generate the formant frequencies (resonances) of speech sounds, an all pole vocal tract model is a very good representation of vocal tract effects for a majority of the speech sounds. Therefore, the vocal tract model can be represented by a set of resonators $(V_k(Z): k=1,\ldots,M)$, with each resonator being an all pole model and M being the number of the desired formant frequencies [30]. In addition, for the purposes of digital simulation, the implementation of V(Z) is not critical. For instance, V(Z) can be represented as a cascade form or a parallel form of $(V_k(Z))$. A cascade form of $V_k(Z)$ as V(Z) is the following:

$$V(Z) = \prod_{k=1}^{M} V_k(Z) = \prod_{k=1}^{M} \frac{1 - 2|Z_k|\cos\theta_k + |Z_k|^2}{1 - 2|Z_k|\cos\theta_k Z^{-1} + |Z_k|^2 Z^{-2}}$$

$$-\prod_{k=1}^{M} \frac{1 - A_k + B_k}{1 - A_k z^{-1} + B_k z^{-2}}$$

$$-\prod_{k=1}^{M} \frac{C_k}{1 - A_k z^{-1} + B_k z^{-2}} \qquad (A7)$$

where $|z_k| = \exp(-\sigma_k T)$

F_k = formant frequencies

T = sampling period

$\sigma_k \simeq$ 1/2 the bandwidth of F_k

$\theta_k = 2\pi F_k T$

$A_k = 2|z_k|\cos\theta_k = 2\exp(-\sigma_k T)\cos(2\pi F_k T)$

$B_k = |z_k|^2 = \exp(-2\sigma_k T)$

$C_k = 1 - A_k + B_k$

Usually, only the first three formant frequencies are sufficient for good quality speech production. Therefore, a commonly used value of M is 3.

Another alternative in implementing V(Z) is by all means using the direct form implementation. Based upon the cascade form of V(Z), the direct form of V(Z) can be realized as follows:

For M=3, the denominator of (A7) can be represented as

$$(1-A_1z^{-1}+B_1z^{-2})(1-A_2z^{-2}+B_2z^{-2})(1-A_3z^{-1}+B_3z^{-2})$$

$$= (1-A_2z^{-1}+B_2z^{-2}-A_1z^{-1}+A_1A_2z^{-2}-A_1B_2z^{-3}+B_1z^{-2}$$
$$-A_2B_1z^{-3}+B_1B_2z^{-4})(1-A_3z^{-1}+B_3z^{-2})$$

$$= (1-(A_1+A_2)z^{-1}+(A_1A_2+B_2+B_1)z^{-2}-(A_1B_2+A_2B_1)z^{-3}$$
$$+B_1B_2z^{-4})(1-A_3z^{-1}+B_3z^{-2})$$

$$= 1-A_3Z^{-1}+B_3Z^{-2}-(A_1+A_2)Z^{-1}+A_3(A_1+A_2)Z^{-2}$$
$$-B_3(A_1+A_2)Z^{-3}+(A_1A_2+B_1+B_2)Z^{-2}$$
$$-A_3(A_1A_2+B_1+B_2)Z^{-3}+B_3(A_1A_2+B_1+B_2)Z^{-4}$$
$$-(A_1B_2+A_2B_1)Z^{-3}+A_3(A_1B_2+A_2B_1)Z^{-4}$$
$$-B_3(A_1B_2+A_2B_1)Z^{-5}+B_1B_2Z^{-4}-A_3B_1B_2Z^{-5}+B_1B_2B_3Z^{-6}$$

Therefore, the direct form of V(Z) is

$$V(Z) = \frac{C_1C_2C_3}{1+V_1Z^{-1}+V_2Z^{-2}+V_3Z^{-3}+V_4Z^{-4}+V_5Z^{-5}+V_6Z^{-6}} \qquad (A8)$$

where $TA1 = A_1 + A_2$; $TA2 = A_1A_2 + B_1 + B_2$

$TA3 = A_1B_2 + A_2B_1$; $TA4 = B_1B_2$

$V_1 = -A_3 - TA1$

$V_2 = A_3 \cdot TA1 + TA2 + B_3$

$V_3 = -B_3 \cdot TA1 - A_3 \cdot TA2 - TA3$

$V_4 = B_3 \cdot TA2 + A_3 \cdot TA3 + TA4$

$V_5 = -B_3 \cdot TA3 - A_3 \cdot TA4$

$V_6 = B_3 \cdot TA4$

As will be shown later, (A8) is used for the direct form realization of the combined model of V(Z) and R(Z), the radiation model.

Radiation model: R(Z)

The whole model up to the vocal tract model V(Z) is good enough for the representation of the air flow from the lungs up the lips. The objective of including the radiation model R(Z) is for the resulting pressure generated by the lips opening (also called as the "radiation impedance"). One approach to obtaining R(Z) is to apply the bilinear

transform to the "radiation impedance", $Z_L(S)$, [30] as follows:

$$R(Z) = Z_L(S)\Big|_{S=\frac{2}{T}\left[\frac{1-Z^{-1}}{1+Z^{-1}}\right]}$$

$$= \frac{SR_rL_r}{R_r + L_rS}\Big|_{S=\frac{2}{T}\left[\frac{1-Z^{-1}}{1+Z^{-1}}\right]} \qquad (A9)$$

where $R_r = \frac{128}{9\pi^2}$; $L_r = \frac{8a'}{3c}$

a' = the radius of the lips opening

c = the velocity of sound
= 35000cm/s

T = sampling period

That is,

$$R(Z) = \frac{\frac{2}{T}\left[\frac{1-Z^{-1}}{1+Z^{-1}}\right]R_rL_r}{R_r + L_r\frac{2}{T}\left[\frac{1-Z^{-1}}{1+Z^{-1}}\right]}$$

$$= \frac{2R_rL_r[1-Z^{-1}]}{TR_r[1+Z^{-1}] + 2L_r[1-Z^{-1}]}$$

$$= \frac{2R_rL_r[1-Z^{-1}]}{[TR_r + 2L_r] + [TR_r - 2L_r]Z^{-1}}$$

$$= \frac{\left[\frac{2R_rL_r}{TR_r + 2L_r}\right][1-Z^{-1}]}{1 + \left[\frac{TR_r - 2L_r}{TR_r + 2L_r}\right]Z^{-1}}$$

$$= \frac{R_1[1-Z^{-1}]}{1 + R_2Z^{-1}} \qquad (A10)$$

where

$$R_1 = \frac{2R_r L_r}{TR_r + 2L_r} = \frac{2\left(\frac{128}{9^2}\right)\left(\frac{8a'}{3c}\right)}{T\left(\frac{128}{9^2}\right) + 2\left(\frac{8a'}{3c}\right)} = \frac{128a'}{24cT + 9^2 a'}$$

$$R_2 = \frac{TR_r - 2L_r}{TR_r + 2L_r} = \frac{T\left(\frac{128}{9^2}\right) - 2\left(\frac{8a'}{3c}\right)}{T\left(\frac{128}{9^2}\right) + 2\left(\frac{8a'}{3c}\right)} = \frac{8cT - 3a'}{8cT + 3a'}$$

Having derived the structure of the speech production model, it is possible to describe how the voiced sounds are produced from this model. In the following, the method used for fixed pitch voiced sounds production is discussed.

Fixed pitch voiced sounds production

If the pitch period for all the voiced sounds of interest is a constant, the model requires only a common excitation source for all the voiced sounds. For the ease of programming the model in a digital computer, the excitation signal of fixed pitch period is generated once and is used for all the voiced sound production. Additionally, the terminal model (V(Z) and R(Z)) is combined into a single time-varying filter, L(Z)=V(Z)·R(Z). From (A8) and (A10), the direct form implementation of L(Z) is the following:

$$L(Z) = \frac{R_1[1 - Z^{-1}]}{1 + R_2 Z^{-1}} \cdot \frac{C_1 C_2 C_3}{1 + V_1 Z^{-1} + V_2 Z^{-2} + V_3 Z^{-3} + V_4 Z^{-4} + V_5 Z^{-5} + V_6 Z^{-6}}$$

$$= [L(1-z^{-1})] \,/\, [1 + (R_2+U_1)z^{-1} + (R_2U_1+U_2)z^{-2} + (R_2U_2+U_3)z^{-3} + (R_2U_3+U_4)z^{-4} + (R_2U_4+U_5)z^{-5} + (R_2U_5+U_6)z^{-6} + R_2U_6z^{-7}]$$

$$= \frac{L(1-z^{-1})}{1+L_1z^{-1}+L_2z^{-2}+L_3z^{-3}+L_4z^{-4}+L_5z^{-5}+L_6z^{-6}+L_7z^{-7}} \qquad (A11)$$

where

$$L = R_1 \cdot C_1 \cdot C_2 \cdot C_3 \; ; \; L_1 = R_2 + U_1 \; ; \; L_2 = R_2 \cdot U_1 + U_2$$

$$L_3 = R_2 \cdot U_2 + U_3 \; ; \; L_4 = R_2 \cdot U_3 + U_4 \; ; \; L_5 = R_2 \cdot U_4 + U_5$$

$$L_6 = R_2 \cdot U_5 + U_6 \; ; \; L_7 = R_2 \cdot U_6$$

The time-varying parameters of L(Z) are, then, the three formant frequencies and the radius of the lips opening. It should be emphasized that the speech production model discussed here was derived especially for the study of GV-VPCM. A more complicated model is required for general speech production of good quality [30].

APPENDIX B

STATISTICAL MEASUREMENT OF SPEECH SIGNALS

The log energy, LOGE, and the unit delay normalized autocorrelation coefficient, r(1), are well-known measures for the classification of speech signals. It has been found [23] that LOGE is excellent for silence discrimination and, r(1) is very good in distinguishing voiced speech segments from unvoiced speech segments.

For each speech segment, define $s(n)$, $n=1,\ldots,N$, to be the n^{th} sample in the segment. LOGE and r(1) are, then, defined as

$$\text{LOGE} = 10*\log_{10}\left[\epsilon + \frac{1}{N}\sum_{n=1}^{N} s^2(n)\right] \quad \text{and,} \tag{B1}$$

$$r(1) = \frac{\sum_{n=1}^{N} s(n)s(n+1)}{\sum_{n=1}^{N} s^2(n)} \tag{B2}$$

where ϵ is a small positive constant added to prevent the computing of the log of zero. The value of ϵ is not critical provided that it is much smaller than the mean-squared value of the speech samples. For our speech signal, the speech samples ranged between $\pm$ 32768. Also, ϵ was set equal to 10^{-4}.

Generally speaking, the LOGE of voiced sounds is much higher than that of silence speech. The LOGE of unvoiced sounds is usually lower than for voiced sounds, but often higher than for silence speech.

The parameter r(1), which varies between -1 and 1, is the correlation between adjacent speech samples. As the energy of voiced sounds is concentrated below about 3 kHz [30] because of the spectrum fall-off introduced by the glottal wave, adjacent samples of voiced speech segments are highly correlated. Hence, the values of r(1) for voiced sounds are close to 1. For unvoiced sounds, most of the energy is found at higher frequencies and their waveform envelopes are noise-like signals. The values of r(1) for unvoiced sounds are close to zero.

The training sequence described in section 3.2.4 is analysed using both LOGE and r(1). The two-dimensional histogram of the LOGE and the r(1), based on speech segments of 10ms, of the training sequence is shown in Table B.1. The silence discrimination using LOGE is firstly described. Subsequently, the boundary values of r(1) for the classifier in AG-VPCM is discussed.

Silence discrimination

Obviously, silence speech segments are with the lowest LOGE compared to voiced and unvoiced speech segments. Hence, silence speech can be distinguished from voiced and unvoiced sounds using the LOGE measure. Referring to Table B.1, the LOGE of the lowest energy speech segments ranged between 10 dB and 20 dB. Therefore, these speech segments correspond to silence speech. However, a LOGE of 20 dB is not yet the boundary value for silence discrimination. One must check for the existence of the silence speech segments when LOGE

is higher than 20 dB. If there are many silence speech segments with a LOGE >20 dB, an appropriate boundary value for silence discrimination will also be higher than 20 dB.

As a result, a boundary value of 30 dB was obtained. It was chosen such that silence speech segments were mostly rejected while unvoiced speech segments were restored. After the silence discrimination (with a boundary value of 30 dB), only 1 silence speech segment was found over 100 speech segments under examination and, the LOGE of all the examined unvoiced speech segments was around 40 dB. A restored silence speech segment (with a LOGE of 31.16 dB) is shown in Fig. B.1 (the first segment). Some of the restored unvoiced speech segments are also shown in Fig. B.2 (the first three segments) and Fig. B.3 (the first two segments).

Boundary values of r(1)

Recall (section 3.7.1, 3.7.2) that the objective of the classifier in AG-VPCM is to separate the voiced sounds with smooth waveforms from the voiced sounds with noise-like waveforms including unvoiced sounds. A classifier using only the r(1) to achieve this requirement is now given.

From Table B.1, only the speech segments with LOGE ranging between 30 dB and 60 dB include those segments with negative r(1). As the r(1) of unvoiced sounds are usually close to zero, it is assumed that the LOGE of the unvoiced speech segments (from the training sequence) is between 30 dB and 60 dB. Additionally, within this range of LOGE, the r(1) of most of the speech segments is higher than 0.94.

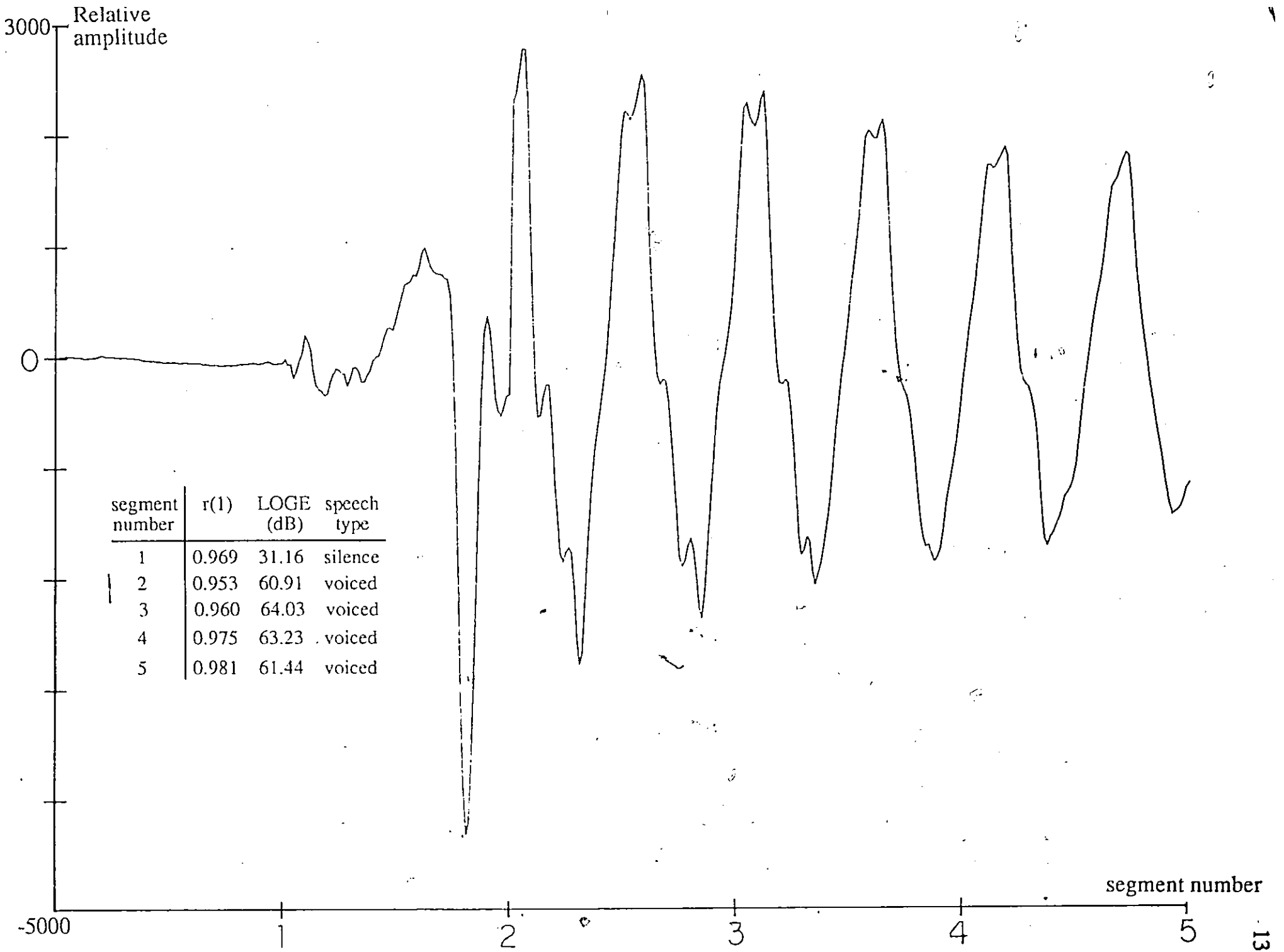
Hence, these speech segments (with a r(1) close to 1) correspond to voiced sounds. About the speech segments with the highest LOGE (>60 dB), the r(1) of most of the speech segments is higher than 0.7. Hence, these speech segments (LOGE >60 dB and r(1) >0.7) correspond to voiced sounds.

According to these voiced/unvoiced judgements, the boundary values of r(1) were set to 0.94 and 0.7. Class 1 (r(1) $\geq$0.94) consists of the voiced sounds with smooth waveforms (eg. the last three segments in Fig. B.1). Class 3 (0.7 >r(1)) consists of the voiced sounds with noise-like waveforms (eg. the last three segments in Fig.B.3) including unvoiced sounds. Class 2 (0.94 >r(1) $\geq$0.7), the intermediate statistical class, consists of the voiced sounds of moderate noisy waveforms (eg. the last two segments in Fig. B.2) and the higher correlated unvoiced sounds (eg. the first three segments in Fig. B.2).

As the end of this appendix, examples (promised in section 3.7.1) about the values of r(1) of voiced sounds with noise-like waveforms and unvoiced sounds are described. The values of the r(1) of voiced and unvoiced sounds were calculated. These results are also shown in Fig. B.1-B.3. From Fig. B.3, the r(1) of the first two segments (unvoiced) are 0.646 and 0.616 and, the r(1) of the last two segments (voiced) are 0.634 and 0.654. Hence, the r(1) of the voiced sounds with noise-like waveforms is close to that of unvoiced sounds. This is also supported by the values of r(1) shown in Fig. B.2.

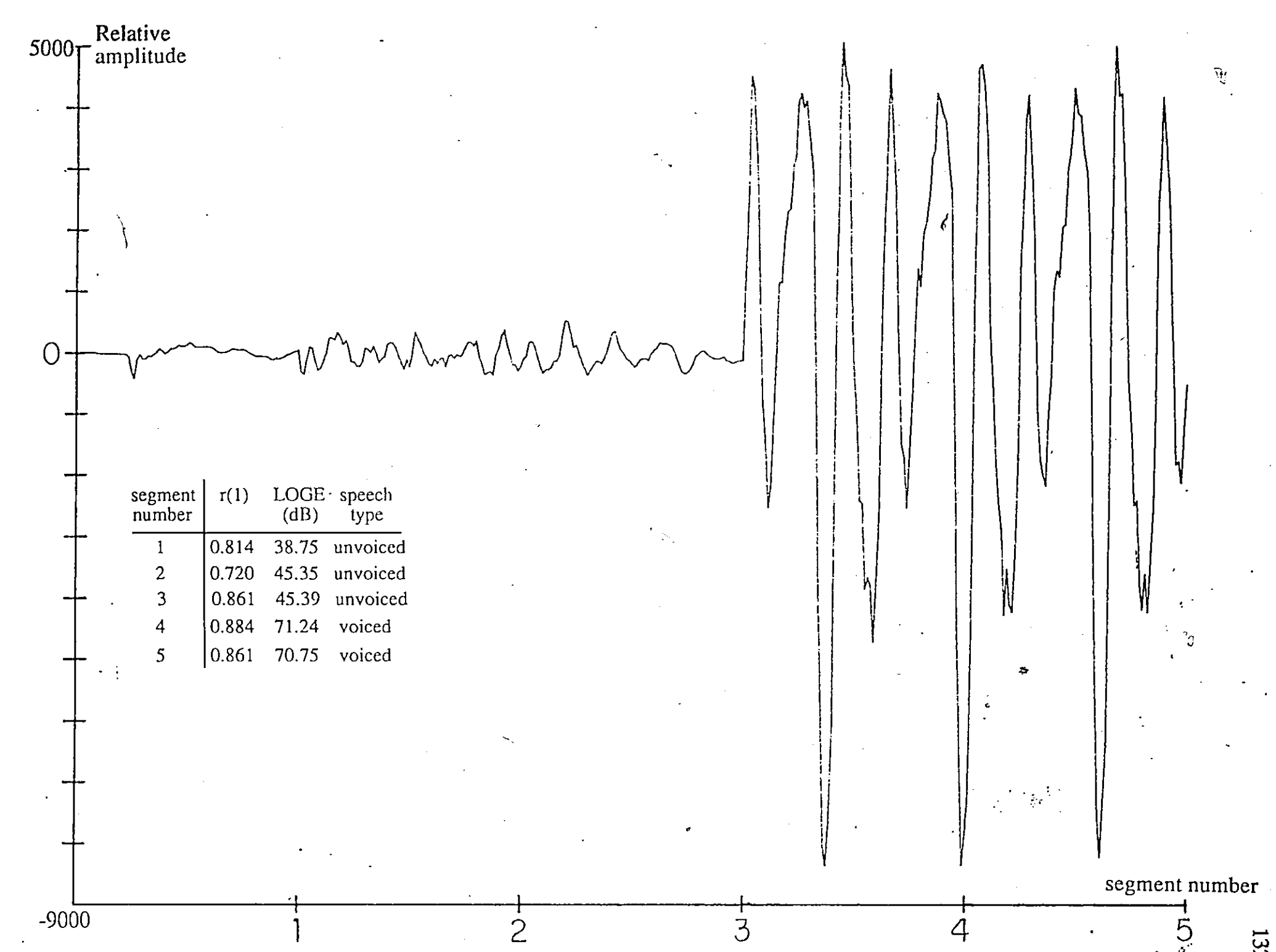
# of speech segment		range of r (1)													
		(1, 0.98]	(0.98, 0.96]	(0.96, 0.94]	(0.94, 0.9]	(0.9, 0.8]	(0.8, 0.7]	(0.7, 0.6]	(0.6, 0.5]	(0.5, 0.4]	(0.4, 0.3]	(0.3, 0.2]	(0.2, 0.1]	(0.1, 0.0]	<0.0
range of LOGE (dB)	[0,10)														
	[10,20)					3	4	1							
	[20,30)	8	97	116	134	106	29	13	6	1					
	[30,40)	328	412	101	42	42	18	13	15	23	20	16	20	7	30
	[40,50)	141	89	11	14	29	32	30	15	13	18	16	16	23	64
	[50,60)	100	126	56	17	14	8	1	1			1	1		6
	[60,70)	44	197	154	267	365	138	40	5	4	1				
	[70,80)			2	38	87	27	3							
	[80,90)														
	[90,100)														

Table B.1 Two-dimensional histogram of r(1) and LOGE (dB)
(based upon speech segments of 10ms)



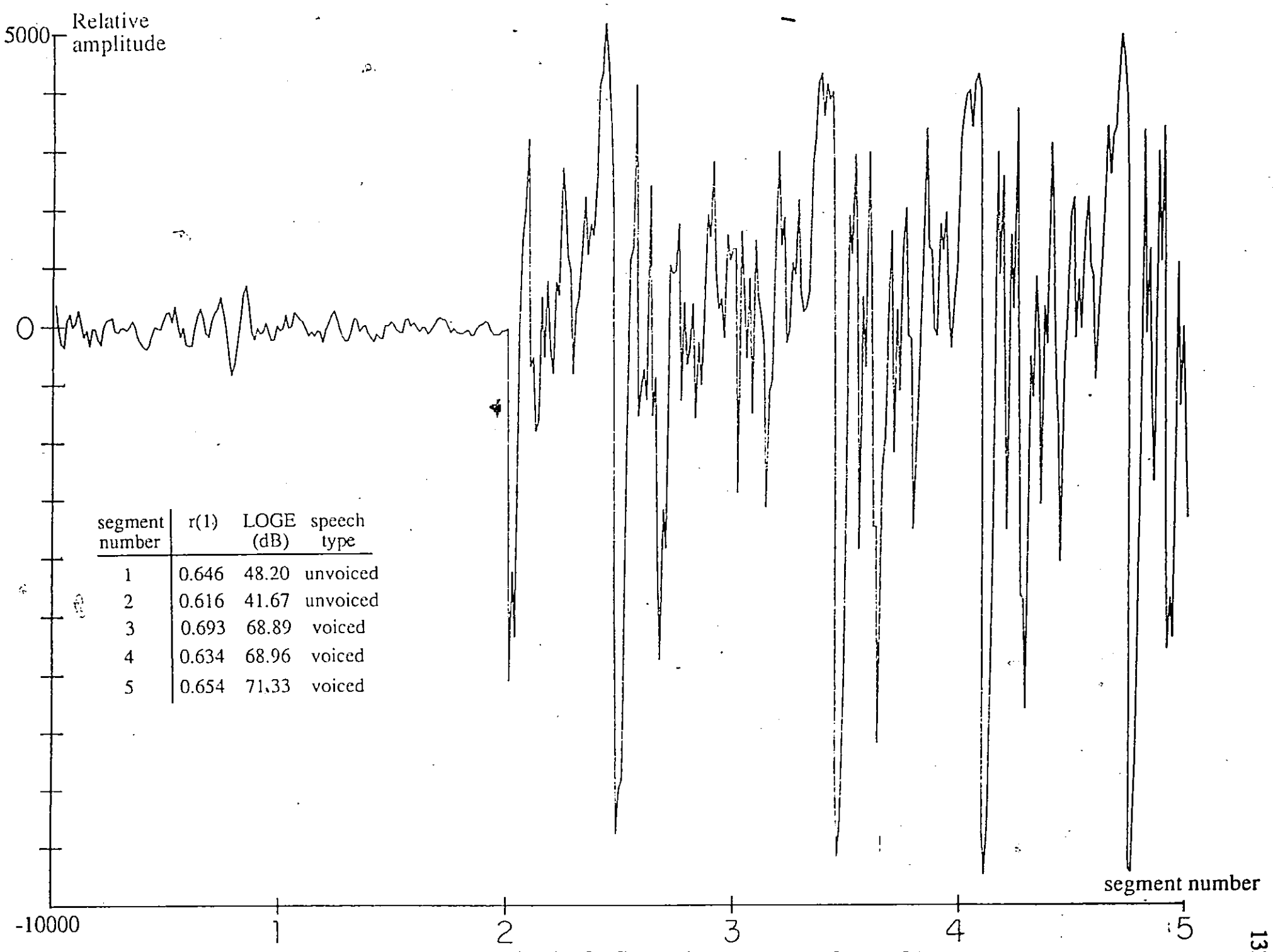
segment number	r(1)	LOGE (dB)	speech type
1	0.969	31.16	silence
2	0.953	60.91	voiced
3	0.960	64.03	voiced
4	0.975	63.23	voiced
5	0.981	61.44	voiced

Fig. B.1 Speech segments from Class 1



segment number	r(1)	LOGE (dB)	speech type
1	0.814	38.75	unvoiced
2	0.720	45.35	unvoiced
3	0.861	45.39	unvoiced
4	0.884	71.24	voiced
5	0.861	70.75	voiced

Fig. B.2 Speech segments from Class 2



segment number	r(1)	LOGE (dB)	speech type
1	0.646	48.20	unvoiced
2	0.616	41.67	unvoiced
3	0.693	68.89	voiced
4	0.634	68.96	voiced
5	0.654	71.33	voiced

Fig. B.3 Speech segments from Class 3

```
C     ********************
C     *   PROGRAM # 1    *
C     *                  *
C     * CODEBOOK DESIGN  *
C     ********************
      COMMON /CPATT/ PD,VD,ER,SPLT
      DIMENSION VD(38210,8),ER(38210,8),PD(1024,8),FK(3821)
      INTEGER VVV(8)
C  This program is able to design the codewords both for VPCM and G-VPCM.
C  When NA>0, the program designs the codewords for VPCM.
C  When NA<0, the program designs the codewodes for G-VPCM.
C  define system
C  LT1 : length (in samples) of the sequence from the first speaker
C  LT2 : length (in samples) of the sequence from the second speaker
C  LT3 : length (in samples) of the sequence from the third speaker
C  LT4 : length (in samples) of the sequence from the fourth speaker
C  kk  : the vector dimension
C  NN  : the number of the required codewords is 2**NN
C  EPS : the distortion conversence threshold
      WRITE(1,*) '? NA,LT1,LT2,LT3,LT4,NN'
      READ(8,*) NA,LT1,LT2,LT3,LT4,NN
      KK=8
      EPS=0.005
      LL1=LT1/KK
      LL2=LT2/KK
      LL3=LT3/KK
      LL4=LT4/KK
      LT=LT1+LT2+LT3+LT4
      LL=LT/KK
      DO 499 I=1,LL1
         READ(3,599) (VVV(KKK),KKK=1,8)
         DO 498 KKK=1,8
         VD(I,KKK)=FLOAT(VVV(KKK))
498      CONTINUE
499   CONTINUE
      I=LL1
      DO 497 J=1,LL2
         READ(4,599) (VVV(KKK),KKK=1,8)
         I=I+1
         DO 496 KKK=1,8
         VD(I,KKK)=FLOAT(VVV(KKK))
496      CONTINUE
497   CONTINUE
      I=LL1+LL2
      DO 495 J=1,LL3
         READ(5,599) (VVV(KKK),KKK=1,8)
         I=I+1
         DO 494 KKK=1,8
         VD(I,KKK)=FLOAT(VVV(KKK))
494      CONTINUE
495   CONTINUE
      I=LL1+LL2+LL3
      DO 493 J=1,LL4
         READ(6,599) (VVV(KKK),KKK=1,8)
         I=I+1
         DO 492 KKK=1,8
         VD(I,KKK)=FLOAT(VVV(KKK))
492      CONTINUE
493   CONTINUE
598   FORMAT(8(1X,F11.4))
```

```
  599  FORMAT(8(I6))
       LLT=10
       LLS=LL/LLT
C  if necessary, normalizes the input training sequence
C  on the basis of segment by segment
       IF(NA.GT.0) GOTO 1111
       DO 10 KLS=1,LLS
          LP=(KLS-1)*LLT
          LLP=LP
          PMA=-1.E18
          DO 20 KLL=1,LLT
             LP=LP+1
             DO 40 KKK=1,KK
                TVD=ABS(VD(LP,KKK))
                IF(TVD.LT.PMA) GOTO 40
                PMA=TVD
  40         CONTINUE
  20       CONTINUE
           FK(KLS)=PMA
           DO 50 KLL=1,LLT
              LLP=LLP+1
              DO 60 KKK=1,KK
                 VD(LLP,KKK)=VD(LLP,KKK)/PMA
  60         CONTINUE
  50       CONTINUE
  10    CONTINUE
C  computes the splitting parameter
 1111 VV=0.0
       DO 70 I=1,LL
          SS=0.0
          DO 80 KKK=1,KK
             SS=SS+(VD(I,KKK))**2
  80      CONTINUE
          VV=VV+SS
  70    CONTINUE
       VAR=VV/(LL*KK)
       SPLT=SQRT(VAR)/10000.0
       NNN=2**NN
C  enter the subroutine for codebook design
       CALL PATT (KK,EPS,LL,NNN,DISTT)
       DO 30 KNN=1,NNN
          WRITE(7,598) (PD(KNN,KKK),KKK=1,KK)
  30   CONTINUE
C  calculate the signal to quantization noise ratio
C  with respect to the training sequence.
       VART=0.0
       DISTS=0.0
       DO 100 KLS=1,LLS
          IF(NA.LT.0) GOTO 2222
          FK(KLS)=1.0
 2222     LP=(KLS-1)*LLT
          DAA=0.0
          DDD=0.0
          DO 200 KLL=1,LLT
             LP=LP+1
             DA=0.0
             DD=0.0
             DO 300 KKK=1,KK
                DA=DA+(VD(LP,KKK)*FK(KLS))**2
                DD=DD+(ER(LP,KKK)*FK(KLS))**2
```

```
  300         CONTINUE
              DAA=DAA+DA
              DDD=DDD+DD
  200       CONTINUE
            VART=VART+DAA
            DISTS=DISTS+DDD
  100   CONTINUE
        DISTM=10.*ALOG10(VART/DISTS)
        WRITE(7,99) DISTM
  99    FORMAT(3X,'THE GFSVQ SQNR=',F6.2)
        STOP
        END
C
C  This subroutine computes the required codewords using
C  the LBG algorithm together with
C  the splitting initialization technique.
C
        SUBROUTINE PATT(KK,EPS,LL,NNN,DISTT)
        COMMON /CPATT/ PD,VD,ER,SPLT
        DIMENSION VD(38210,8),ER(38210,8),PD(1024,8)
        INTEGER PROB(1024)
        DIMENSION ID(38210),TD(1024,8),DEV(1024)
C  computes the centroid of the training sequence
        DO 100 KKK=1,KK
           TD(1,KKK)=0.0
           DO 101 KLL=1,LL
              TD(I,KKK)=TD(1,KKK)+VD(KLL,KKK)
  101      CONTINUE
           TD(1,KKK)=TD(1,KKK)/LL
  100   CONTINUE
C  initializes algorithm
        MM=1
  170   MM=MM*2
        DIST1=1.E18
C  terminates the algorithm if the required codebook is found
C  otherwise doubles the current codewords by splitting
        IF(MM.GT.NNN) GOTO 171
        DO 111 KMM=1,MM,2
           KMMT=(KMM+1)/2
           DO 112 KKK=1,KK
              PD(KMM,KKK)=TD(KMMT,KKK)
              PD(KMM+1,KKK)=TD(KMMT,KKK)-SPLT
  112      CONTINUE
  111   CONTINUE
C
C       The LBG algorithm
C
C  find the set of partitions
  172   DO 113 KLL=1,LL
           DO 114 KNN=1,MM
              DEV(KNN)=0.0
  114      CONTINUE
           DO 115 KNN=1,MM
              DO 116 KKK=1,KK
                 DEV(KNN)=DEV(KNN)+(VD(KLL,KKK)-PD(KNN,KKK))**2
  116         CONTINUE
  115      CONTINUE
           DIST=1.E18
           KNN=0
  134      KNN=KNN+1
```

```
          IF(KNN.GT.MM) GOTO 135
          IF(DEV(KNN).GT.DIST) GOTO 134
            DIST=DEV(KNN)
            KIST=KNN
            GOTO 134
  135     ID(KLL)=KIST
  113   CONTINUE
C   calculate the sample average distortion
        DISTT=0.0
        DO 117 KLL=1,LL
           S=0.0
           IDN=ID(KLL)
           DO 118 KKK=1,KK
              S=S+(VD(KLL,KKK)-PD(IDN,KKK))**2
  118      CONTINUE
           DISTT=DISTT+S
  117   CONTINUE
        DISTT=DISTT/LL
        IF(DISTT.EQ.0.0) GOTO 173
        IF(ABS((DIST1-DISTT)/DISTT).LT.EPS) GOTO 170
          DIST1=DISTT
C   find the optimal codebook
  173   DO 119 KNN=1,MM
           DO 120 KKK=1,KK
              PD(KNN,KKK)=0.0
  120      CONTINUE
           PROB(KNN)=0
           DO 121 KLL=1,LL
              IF(ID(KLL).NE.KNN) GOTO 121
              PROB(KNN)=PROB(KNN)+1
              DO 122 KKK=1,KK
                 PD(KNN,KKK)=PD(KNN,KKK)+VD(KLL,KKK)
  122         CONTINUE
  121      CONTINUE
           IF(PROB(KNN).EQ.0) GOTO 136
           DO 123 KKK=1,KK
              PD(KNN,KKK)=PD(KNN,KKK)/PROB(KNN)
              TD(KNN,KKK)=PD(KNN,KKK)
  123      CONTINUE
           GOTO 119
C   keep the old partition if the new partition is empty
  136      KM=(KNN+1)/2
           DO 124 KKK=1,KK
              PD(KNN,KKK)=TD(KM,KKK)
  124      CONTINUE
  119   CONTINUE
        IF(DISTT.EQ.0.0) GOTO 170
        GOTO 172
C   the codebook design is completed.
C   calculate the error waveform that results from
C   reconstructing the training sequence
  171   DISTS=0.0
        DO 125 I=1,LL
           DE=0.0
           DO 126 KKK=1,KK
              DE=DE+(VD(I,KKK)-PD(1,KKK))**2
  126      CONTINUE
           DMINI=DE
           IDE=DMINI
           DO 127 N=2,NNN
```

```
            DE=0.0
            DO 128 KKK=1,KK
               DE=DE+(UD(I,KKK)-PD(N,KKK))**2
128         CONTINUE
            IF(DE.GT.DMINI) GOTO 127
            DMINI=DE
            IDE=N
127      CONTINUE
         IF(DMINI.NE.IDE) GOTO 22
         IDE=1
22       DO 142 KKK=1,KK
            ER(I,KKK)=UD(I,KKK)-PD(IDE,KKK)
142      CONTINUE
125   CONTINUE
      RETURN
      END
```

```
C  ******************************
C  *       PROGRAM # 2          *
C  *                            *
C  * THE REALIZATION OF AG-VPCM *
C  ******************************
      DIMENSION VD(38210,8),TD(80),FK(3821),TTD(8),RDN(8),RD(38210,8)
      DIMENSION PPD(256,8),PD1(256,8),PD2(256,8),PD3(256,8),R(3821)
      REAL     MSNR
      INTEGER VVV(8)
C  This program realizes both A-VPCM and AG-VPCM
C  When NA>0, the program realizes A-VPCM
C  When NA<0, the program realizes AG-VPCM.
C  define system
C  PD1 : the codebook of Class 1 (r(1),se,0.94)
C  PD2 : the codebook of Class 2 (r(1),se,0.7 and r(1),lt,0.94)
C  PD3 : the codebook of Class 3 (r(1),lt,0.7)
C  LT  : length (in samples) of the test sequence
C  KK  : the vector dimension
C  N   : the number of codewords in each codebook
      WRITE(1,*) ' NA,LT '
      READ(2,*) NA,LT
      R1T=0.94
      R2T=0.7
      N=256
      KK=8
      LL=LT/KK
      DO 499 I=1,LL
         READ(12,598) (VVV(KKK),KKK=1,8)
         DO 498 KKK=1,8
         VD(I,KKK)=FLOAT(VVV(KKK))
  498    CONTINUE
  499 CONTINUE
      DO 497 I=1,N
         READ(13,599) (PD1(I,KKK),KKK=1,KK)
  497 CONTINUE
      DO 496 I=1,N
         READ(14,599) (PD2(I,KKK),KKK=1,KK)
  496 CONTINUE
      DO 495 I=1,N
         READ(15,599) (PD3(I,KKK),KKK=1,KK)
  495 CONTINUE
  599 FORMAT(8(1X,F11.4))
  598 FORMAT(8(I6))
      LLT=10
      LLF=LLT*KK
      LLS=LL/LLT
      DO 10 KLS=1,LLS
         LP=(KLS-1)*LLT
         LLP=LP
         PMA=-1.E18
         DO 20 KLL=1,LLT
            LP=LP+1
            KL=(KLL-1)*KK
            DO 30 KKK=1,KK
               TD(KL+KKK)=VD(LP,KKK)
  30        CONTINUE
            DO 31 KKK=1,KK
               TVD=ABS(VD(LP,KKK))
               IF(TVD.LT.PMA) GOTO 31
               PMA=TVD
```

```
 31          CONTINUE
 20       CONTINUE
          FK(KLS)=PMA
C  if necessary, normalizes the test sequence
C  on the basis of segment by segment
          IF(NA.GT.0) GOTO 1111
          DO 21 KLL=1,LLT
             LLP=LLP+1
             DO 22 KKK=1,KK
                VD(LLP,KKK)=VD(LLP,KKK)/PMA
 22          CONTINUE
 21       CONTINUE
C  classifies the current speech segment
C  into one of the three different classes and,
C  assigns the appropriate codebook for the segment
 1111     TS=TD(1)
          R0=TS**2
          R1=0.0
          DO 40 KLL=2,LLF
             R0=R0+(TD(KLL))**2
             R1=R1+TD(KLL)*TS
             TS=TD(KLL)
 40       CONTINUE
          R(KLS)=R1/R0
          IF(R(KLS).GE.R1T) GOTO 51
          IF(R(KLS).GE.R2T) GOTO 52
          DO 50 I=1,N
             DO 60 J=1,KK
                PPD(I,J)=PD3(I,J)
 60          CONTINUE
 50       CONTINUE
          GOTO 53
 51       DO 70 I=1,N
             DO 80 J=1,KK
                PPD(I,J)=PD1(I,J)
 80          CONTINUE
 70       CONTINUE
          GOTO 53
 52       DO 90 I=1,N
             DO 100 J=1,KK
                PPD(I,J)=PD2(I,J)
 100         CONTINUE
 90       CONTINUE
C  vector quantizes the current speech segment
 53       LP=(KLS-1)*LLT
          DO 110 KLL=1,LLT
             LP=LP+1
             DO 120 KKK=1,KK
                TTD(KKK)=VD(LP,KKK)
 120         CONTINUE
C  do the vector quantization
             CALL FSER(N,TTD,PPD,RDN)
             DO 130 KKK=1,KK
                RD(LP,KKK)=RDN(KKK)
 130         CONTINUE
 110      CONTINUE
 10    CONTINUE
C  computes the SNR and the SEGSNR
       CALL SNR(NA,LL,VD,RD,FK,MSNR,SSNR)
       WRITE(1,*) 'SNR=',MSNR, 'SEGSNR=',SSNR
```

```
      STOP
      END
C
C  This subroutine realizes the
C  full search vector quantizer
C
      SUBROUTINE FSER (NM,TTD,TPD,RDD)
      DIMENSION TTD(8),TPD(256,8),RDD(8)
      K=8
      DEV=0.0
      DO 10 I=1,K
         DEV=DEV+(TTD(I)-TPD(1,I))**2
 10   CONTINUE
      TEMP=DEV
      TTMC=TEMP
      DO 20 I=2,NM
         DEV=0.0
         DO 30 J=1,K
            DEV=DEV+(TTD(J)-TPD(I,J))**2
 30      CONTINUE
         IF(DEV.GT.TEMP) GOTO 20
         TEMP=DEV
         IN=I
 20   CONTINUE
      IF(TEMP.NE.TTMC) GOTO 11
      IN=1
 11   DO 40 I=1,K
         RDD(I)=TPD(IN,I)
 40   CONTINUE
      RETURN
      END
C
C  This subroutine computes the SNR and the SEGSNR
C
      SUBROUTINE SNR (NA,LL,VD,RPD,FK,MSE,SEG)
      DIMENSION VD(38210,8),RPD(38210,8),FK(38210),RRD(38210,8)
      REAL      MSE
      K=8
      LLT=10
      LLS=LL/LLT
      VARR=0.0
      DISEE=0.0
      SENR=0.0
      DO 10 KLS=1,LLS
         IF(NA.LT.0) GOTO 11
         FK(KLS)=1.0
 11      I=(KLS-1)*LLT
         VARE=0.0
         DISTE=0.0
         DO 20 KLL=1,LLT
            I=I+1
            SV=0.0
            DEE=0.0
            DO 30 KK=1,K
               SV=SV+(VD(I,KK)*FK(KLS))**2
               DEE=DEE+((VD(I,KK)-RPD(I,KK))*FK(KLS))**2
               RRD(I,KK)=RPD(I,KK)*FK(KLS)
 30         CONTINUE
            DISTE=DISTE+DEE
            VARE=VARE+SV
```

```
 20       CONTINUE
          SENRL=10.*ALOG10(VARE/DISTE)
          SENR=SENR+SENRL
          DISER=DISER+DISTE
          VARR=VARR+VARE
 10    CONTINUE
       SEG=SENR/LLS
       MSE=10.*ALOG10(VARR/DISFE)
       DO 40 I=1,LL
          WRITE(6,599) (RRD(I,KK),KK=1,K)
 40    CONTINUE
 599   FORMAT(8(1X,F11.4))
       RETURN
       END
```

REFERENCES

[1] R. M. Gray, "Vector Quantization", IEEE ASSP Magazine, pp.4-28, April 1984.

[2] J. Makhoul, S. Roucos and H. Gish, "Vector Quantization in Speech Coding", Proc. of the IEEE, vol.73, pp.1551-1588, November 1985.

[3] A. Gersho and Cuperman, "Vector Quantization: A Pattern-Matching Technique for Speech Coding", IEEE Commun. Magazine, pp.15-21, December 1983.

[4] A. Buzo, A. H. Gray Jr., R. M. Gray and J. D. Markel, "Speech Coding Based Upon Vector Quantization", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-28 pp.562-574, October 1980.

[5] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design", IEEE Trans. on Commun., vol. COM-28, pp.84-95, January 1980.

[6] H. Abut, R. M. Gray and G. Rebolledo, "Vector Quantization of Speech and Speech-Like Waveforms", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-30, pp.423-435, June 1982.

[7] R. M. Gray, and Y. Linde, "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources", IEEE Trans. on Commun., vol. COM-30, pp.381-389, February 1982.

[8] G. Rebolledo, R. M. Gray and J. P. Burg, "A Multirate Voice Digitizer Based Upon Vector Quantization", IEEE Trans. Commun., vol. COM-30, pp.721-727, April 1982.

[9] D. Y. Wong, B. H. Juang and A. H. Gray Jr., "An 800 bits/s Vector Quantization LPC Vocoder", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-30, pp.770-780, October 1982.

[10] M. J. Sabin and R. M. Gray, "Product Code Vector Quantizers for Waveform and Voice Coding", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-32, pp.474-488, June 1984.

[11] M. Dunham and R. M. Gray, "An Algorithm for the Design of Labeled-Transition Finite-State Vector Quantizers", IEEE Trans. on Commun.,vol. COM-33, pp.83-89, January 1985.

[12] R. M. Gray, A. Buzo, A. H. Gray Jr. and Y. Matsuyama, "Distortion Measures for Speech Processing", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-28, pp.367-376, August 1980.

[13] A. Gersho, "Asymptotically Optimal Block Quantization", IEEE Trans. on Information Theory, vol. IT-25, pp.373-380, July 1979.

[14] A. Gersho, "On the Structure of Vector Quantizers", IEEE Trans. on Information Theory, vol. IT-28, pp.157-166, March 1982.

[15] D. Wolf, H. Reginінger and J. Dziumbla, "A comparative study of various speech encoding schemes using vector quantization", ICASSP, pp.10.8.1-10.8.4, 1984.

[16] N. S. Jayant and P. Noll, Digital Coding of Waveforms. Englewood Cliffs, NJ: Prentice-Hall, 1984

[17] J. Bellamy, Digital Telephony. John Wiley & Sons, 1982.

[18] M. Copperi and D. Sereno, "Vector Quantization and Perceptual Criteria for Low-Rate Coding of Speech", ICASSP, pp.7.6.1-7.6.4, 1985.

[19] S. Roucos, R. Schwartz and J. Makhoul, "Vector Quantization for Very-Low-Rate Coding of Speech", IEEE Global Telecommun. Conf., pp.E6.2.1-E6.2.5, 1982.

[20] H. Reininger and D. Wolf, "Speech and Speaker Independent Codebook Design in VQ Coding Schemes", ICASSP, pp.43.8.1-43.8.3, 1985.

[21] J. Makhoul and M. Berouti, "Adaptive Noise Spectral Shaping and Entropy Coding in Predictive Coding of Speech", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-27, pp.63-72, February 1979.

[22] V. Cuperman and A. Gersho, "Vector Predictive Coding of Speech at 16 kbits/s", IEEE Trans. on Commun., vol. COM-33, pp.685-696, July 1985.

[23] B. S. Atal and L. R. Rabiner, "A Pattern Recognition Approach to Voiced-Unvoiced-Silence Classification with Applications on Speech Recognition", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-24, pp.201-212, June 1976.

[24] B. P. M. Tao, H. Abut and R. M. Gray, "Hardware Realization of Waveform Vector Quantizers", IEEE Journal on Selected Areas in Commun., vol. SAC-2, pp.343-352, March 1984.

[25] G. Davidson, T. Stanhope, R. Aravind and A. Gersho, "Real-Time Speech Compression with a VLSI Vector Quantization Processor", ICASSP, pp.37.6.1-37.6.4, 1985.

[26] D. Y. Cheng, A. Gersho, B. Ramamurthi and Y. Shoham, "Fast Search Algorithm for Vector Quantization and Pattern Matching", ICASSP, pp.9.11.1-9.11.4, 1984.

[27] R. W. Schafer and L. R. Rabiner, "Digital Representations of Speech Signals", Proc. of the IEEE, vol.63, pp.662-677, April 1975.

[28] J. Makhoul, "Linear Prediction: A Tutorial Review", Proc. of the IEEE, vol.63, pp.561-580, April 1975.

[29] R. Viswanathan and J. Makhoul, "Quantization Properties of Transmission Parameters in Linear Predictive Systems", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-23, pp.309-321, June 1975.

[30] L. R. Rabiner and R. W. Schafer, Digital Processing of Speech Signals. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[31] P. V. S. Rao and R. B. Thosar, "A Programming System for Speech Synthesis", IEEE Trans. on Acoust., Speech and Signal Processing, vol. ASSP-22, pp.217-227, June 1974.